AUCCCR: Agent Utility Centered Clustering for Cooperation Recommendation

Amaury Bouchra Pilet, Davide Frey and François Taïani

5 Germinal 229

IRISA, Inria, Univ Rennes, CNRS

Abstract

Providing recommendation to agents (e.g. people or organizations) regarding whom they should collaborate with in order to reach some objective is a recurring problem in a wide range of domains. It can be useful for instance in the context of collaborative machine learning, grouped purchases, and group holidays. This problem has been modeled by hedonic games, but this generic formulation cannot easily be used to provide efficient algorithmic solutions. In this work, we define a class of hedonist games that allows us to provide an algorithmic solution to the collaboration recommendation problem by means of a clustering algorithm. We evaluate our algorithm, theoretically and experimentally and show that it performs better than other clustering algorithms in this context.

I Introduction

In this work, we aim to provide an algorithmic solution for people or organizations who wish to collaborate towards some task (buying, machine-learning...) but do not want to cooperate with members whose individual objectives are too different from their own. This problem can be modeled as a set of rational agents that may or may not form coalitions depending on the utility they might derive from such coalitions, a formalism based on economics and prescriptive decision theory [vM44]. While descriptive decision theory has shifter to the more exact Prospect theory[KT79], this work is about automated decision support and, thus, based on expected utility[Ran+07]. The utility an agent obtains from belonging to a group is positively correlated to the group's size (larger is better), but negatively correlated with its distance from the group's barycenter (closer is better). Such a model can capture various practical problems: in collaborative machine learning, for instance, learners with similar but different tasks may or may not collaborate with each other, depending on the effect of this collaboration on the efficiency of their learning process; in grouped purchases, potential buyers will search for other people with similar buying habits to save money by placing grouped orders, but will not benefit if the products bought deviates too much from their preferred options; when planing organized vacations, most people want to save money with group rates, but not at the cost of visiting too many places they are not interested in. We also consider the case where people want to be in large enough groups but not too large.

An algorithm that solves this problem should provide collaboration recommendations (e.g. with whom each agent should perform a grouped order) that agents find acceptable and from which they do not deviate. Since, in this context, the relative utility of different options for an agent depends on the choices of other agents, it is crucial to prevent situations where a few agents reject the recommendation, as this could lead to a complete collapse of the solution. The departure of a dissatisfied agent can decrease the group's value for other agents, which may in turn leave the group, etc.

In economic terms, this problem can be modeled as a *hedonic game* [DG80], but this formalization tends unfortunately to be too general to allow for effective algorithmic solutions. Existing algorithms that solve generic hedonic games, summarized in [AS16], need to constrain them by requiring the existence of some kind of equilibrium (Nash equilibrium or a similar definition), which is not guaranteed by the generic definition of hedonic games and may not exist in practice. In our work, we instead consider a more restricted class of games, and we provide a solution that also works in cases where no kind of equilibrium exists.

Hedonic games designed for specific problems have been proposed in the past, for example in [Saa+10], but the practical problems they consider do not apply to our context. In particular, these works tend to adopt a formalization that ensures the existence of some kind of equilibrium, while we want solutions for cases where no equilibrium exists.

Our insight is that the hedonic games corresponding to our coalition-formation problem can be interpreted as a clustering problem. We want to identify groups of close and numerous agents, which is essentially a clustering task. However, commonly used clustering algorithms, while technically applicable, are not adapted to this particular task. To address this gap, this paper proposes a novel clustering algorithm, that is specifically designed to address the task we want to solve.

We begin this paper with a formal definition of the class of hedonic games we use to model collaboration. We then propose AUCCCR (pronounced "okr", IPA: [okʁ]), a clustering algorithm able to provide solutions to the considered problem. We present a theoretical and experimental evaluation of our algorithm on both synthetic and real data. From this evaluation, we conclude that our algorithm respects individual agents interests better than other clustering algorithms.

II Our approach

II.1 Problem statement and formalization

We consider a set of agents $a_0, ..., a_m \in A$ that seek to form groups so that every agent in a group effectively benefits from belonging to this group. We represent agents' preferences using an ℓ-dimensional real vector given by a projector function, $p : A \to \mathbb{R}^\ell$, which allows us to measure similarity between agents as a distance. In our model, the benefit from being in a group depends on two factors: the size of the group (the larger the better), and the similarity between an agent and the (average of) the group (the more similar the better).

More concretely, we define a *utility* function $U_p(a, g)$ that expresses the interest of an agent, a, in a group, g, using a projector, p, as the product of two factors: *(i)* the value of the group (function v), which grows with the group's size; and *(ii)* a decreasing function ($n(...)$) of the distance between an agent and the group's barycenter (as measured by a distance function $d(...)$). This is formally captured by the following formula:

$$U_p(a, g) : A \times \mathcal{P}(A) \to \mathbb{R}^+ = n\Big(d(p(a), bary_p(g))\Big) \times v(\#g),$$

where $\mathcal{P}(A)$ is the power set of A, $\#g$ is the size of group g, $bary_p(g)$ is the barycenter of group g with projector p, and the functions d, v, and n are defined as follows: $d(x, y) : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ is the *distance* between x and y; $v(n) : \mathbb{N} \to [1, +\infty[$ is the *value* of a group of size n (increasing, $v(1) = 1$); and $n(d) : \mathbb{R}^+ \to [0, 1]$ is the *normalizer* for an agent at distance d from its group's barycenter (decreasing, $n(0) = 1$). In the following we will also use $g(a)$ to denote the group of agent a. These definitions associate a utility value of 1 with an agent that remains alone (since the interest of joining a group with a barycenter at distance 0 and consisting of 1 element (itself) is $n(0) \times v(1) = 1 \times 1$).

Given A, $p()$, $d()$, $n()$ and $v()$, the group formation problem we seek to solve consists in finding a partition of A (representing the groups), that maximizes the sum of every agent's utility, while minimizing (or even eliminating) the benefit individual agents could gain by either changing group or

Ident	Description
$rand(A)$	Random (uniform) element of A
$rand(A, d)$	Random element of A with distribution d
$d(x, y)$	distance between x and y
$bary(g)$	barycenter of group g
$p(a)$	projector function
A	set of agents
k	number of clusters
dmin2	k-means++ distribution
grp	groups
ngrp	new groups (in building)
size	groups' sizes
nsize	new groups' sizes (in building)
bgrp	estimated best groups for each agent (inner loop)
val	affectation's value (sum of utilities)
nval	new affectation's (just built) value (sum of utilities)

Table 1: Symbols used in algorithm

remaining alone (to ensure the solution's stability). Table 1 summarizes these notations and defines the variables and parameters appearing in algorithms.

II.2 Algorithm

In this section, we present our algorithm's and define them precisely.

In the following algorithms, the member function $*.invertkv()$ inverts keys and values of an associative container (data structure), the member function $*.group(x)$ returns the set of all elements in the container associated with the same key as x, singleton x if x is not in the container.

Algorithm 1 presents the control function (loop) of our group recommendation algorithm. It loops over the parameter k (number of groups), increasing it until no significant global utility (sum of all agents utility) gains have been achieved. To reduce the randomness of the process (our clustering algorithm being randomized) each k value is tried several times and the loop will only terminate if no gain happen for several k increments (momentum).

Algorithm 2 is based on k-means[Llo82]. The first step is the initialization of groups, which is a k-means++[AV07] initialization (each group is given one single random member sequentially, with probability increasing with the distance to existing groups). After initialization, the process consists of two nested loops (inner and outer). Since, unlike k-means, we take group size into account, we need two loops, one for distance changes (as k-means) and one for group size changes (the inner loop). In the inner loop, at initialization, each group, has a fixed barycenter and is given a potential size of $\#A$. At each turn, agents compute their utility for each group with the potential sizes and choose their group accordingly. After that, the potential size of each group is set to the number of agents choosing this group, the barycenters are not updated. The process is repeated until no change append in one turn. In the outer loop, at each turn, the inner loop is executed and the barycenters are updated. The loop terminates if no change append in one turn.

Compared with k-means, the interesting part of our algorithm is the inner loop. Since, unlike k-means, our algorithm take the size of groups into account, we need this inner loop. It allows us to compute agents' utility with bigger groups. For agents to join groups, these groups have to be big enough and for groups to be big, agents have to join them. Our inner loop, with its over-evaluated groups' initial potential sizes, breaks this circle. The main limitation of this algorithm is that we do not have a theoretical termination proof for it. It could potentially, in very rare cases, get stuck in an infinite loop. So, we propose variation with guaranteed termination in Algorithm 3.

Algorithm 3 terminates the inner loop if the total number of agents member of any group does not

Algorithm 1: Recommendation algorithm

Data: A set of agents, p projector, prc is the number of (random) initial values to try, mmt momentum effect
Result: A pair of a group affectations of agents and its value

```
1  val ← #A
2  k ← 1
3  pmm ← mmt
4  repeat
5      pval ← val
6      val ← #A
7      repeat
8          afect ← CLUSTER(A, p, k)
9          nval ← Σ_{a∈A} n(d(p(a), bary(afect.group(a)))) × v(#afect.group(a))
10         if nval > val then
11             if nval > pval then
12                 sol = afect
13             val ← nval
14     until prc times
15     k++
16     if val > pval then
17         pmm ← mmt
18     else
19         pmm − −
20 until pmm = 0
21 return (sol, pval)
```

change in one turn and the outer loop if the global utility of the affectation does not increase in one turn. This way, the algorithm is guaranteed to converge.

Both algorithms include a somewhat complex formula to compute agent's interest. For large instances, with very big clusters, individual agents influence on a cluster can be considered negligible. We call this situation "agents' atomicity" (agents are "atoms", too small relatively to the whole system to have significant influence individually). Thus, we can make an agents' atomicity hypothesis to simplify the formulas. $n\Big(d(p(a), bary(grp[i] \cup a))\Big) \times v(size[i] + \mathbb{1}_{a \notin grp[i]})$ can be replaced by $n\Big(d(p(a), bary(grp[i]))\Big) \times v(size[i])$. This possibility may be used for a production implementation.

III Theoretical analysis

Here we prove the essential property of the clustering algorithm presented in Algorithm 2.

Property (Nash equilibrium)**.** *If we consider a game where each agent's (player) possible choices are k different groups to join, or not choosing any group, and each agent's a payoff for choosing group g is given by $U_p(a, g)$ as defined before and 1 for no choosing any group, the affectation outputed by Algorithm 2, if it exists, is a (weak) Nash equilibrium for this game.*

Proof (Nash equilibrium)**.** We assume that our algorithm as converged, we consider the last iteration of the loop guarded by $ngrp = grp$ (line 11) and, in this iteration, the last iteration of the loop guarded by $nsize = size$ (line 15).

Algorithm 2: Clustering algorithm

function CLUSTER(A, p, k) **is**

    /* k-means++ initialization */

    $ra \leftarrow rand(A)$

    $ngrp[0] \leftarrow \{ra\}$

    foreach $a \in A$ **do**

        $dmin2[a] \leftarrow d(p(a), p(ra))^2$

    for $1 \leqslant i < k$ **do**

        $na \leftarrow rand(A, dmin2)$

            /* random element from A, relatives probabilities given by $dmin2$ */

        $ngrp[i] \leftarrow \{na\}$ // appends $\{na\}$ to ngrp

        foreach $a \in A$ **do**

            $dmin2[a] \leftarrow \min(d(p(a), p(na))^2, dmin2[a])$

    /* main loop */

    repeat

        $grp \leftarrow ngrp$

        $ngrp.clear()$

        $nsize \leftarrow [\#A...\#A]$

        repeat

            $size \leftarrow nsize$

            $nsize \leftarrow [0...0]$

            foreach $a \in A$ **do**

                $bgrp[a] \leftarrow \text{argmax}_{i \in \bot \cup [\![0, k-1]\!]}(n\Big(d(p(a), bary(grp[i] \cup a))\Big) \times v(size[i] + \mathbb{1}_{a \notin grp[i]}))$

                /* assuming $grp[\bot] = \emptyset$ and $size[\bot] = 0$ */

                $nsize[bgrp[a]] + +$ // does nothing if $bgrp[a] = \bot$

        until $nsize = size$

        $ngrp \leftarrow bgrp.invertkv()$ // key-value inversion

    until $ngrp = grp$

    return $ngrp$

Let's assume that the output of the function, $ngrp = grp$, is not a (weak) Nash equilibrium. Then $\exists a \in A, g \in \bot \cup [\![0, k-1]\!](U_p(a, g) > U_p(a, grp.group(a)))$.
Since at the end of the loop iteration $nsize = size$ and $nsize$ is, by design, the vector of the sizes of the groups given by $ngrp$, we can affirm that, during the considered loop iteration (assimilating i with a group), $U_p(a, grp[i]) = n\Big(d(p(a), bary(grp[i] \cup a))\Big) \times v(size[i] + \mathbb{1}_{a \notin grp[i]})$.
Since, by design, our algorithm selects i to maximize $n\Big(d(p(a), bary(grp[i] \cup a))\Big) \times v(size[i] + \mathbb{1}_{a \notin grp[i]})$, we can affirm that, in this loop iteration, a different group would have been selected for at least one a. Now, either that changes the size of at least one group, implying $nsize \neq size$ at the end of the loop, which is a contradiction. Or, if the sizes remains identical (exchange of members), then, the loop guarded by $nsize = size$ exists, and we get $ngrp \neq grp$, which also is a contradiction.□

Since the existence of a Nash equilibrium for this game is not guaranteed, it is possible that Algorithm 2 does not terminate. Now, we prove that Algorithm 3, for which we do not have an equilibrium proof, will always terminate.

Property (Convergence)**.** *Algorithm 3 always terminates.*

Proof (Convergence)**.** Among the control structures used in Algorithm 3, the two repeat...until loops

are the only that do not trivially terminate (for loops terminate trivially). For the inner loop, we replaced $nsize = size$ by $nsize.sum() \geqslant size.sum()$ as termination condition. This implies that, during the loop, $size.sum()$'s values are a strictly decreasing natural ($\mathbb{N}$) sequence. Such a sequence can not be infinite, thus, the inner loop terminates. For the outer loop, we replaced $ngrp = grp$ by $nval \leqslant val$ as termination condition. This implies that, during the loop, val's values are a strictly increasing real sequence. Moreover, those values are given by a formula taking as parameter an affectation of a finite number of agents in a finite number of groups. The possible inputs for this formula for a given execution of the algorithm (fixed parameters) are a finite set. This implies that the possible values for val (output of this formula) are also a finite set (for a given execution of the algorithm). Thus, val's values are a strictly increasing sequence of elements of a finite set (the order is given by the classical order for real number). Such a sequence can not be infinite, thus, the outer loop terminates. We can now conclude that our algorithm will always terminate.□

We note that Algorithm 3 is not guaranteed to give a Nash equilibrium. Such an equilibrium may simply not exist, but even if it exists, the algorithm is not guaranteed to find it. Due to the end condition of its outer loop, Algorithm 3, while still based on agents' self-interest, is more centered on attaining general optimality (maximizing the sum of all agents utilities) than Algorithm 2.

An important thing that differentiates our algorithm from generic approaches to solve hedonic games presented in [AS16] is the inner loop of our algorithm, which allows groups to grow to a point that is better for every agent even in situations where individual rational decisions could not.

Let's examine this simple example : We have four agents, A, B, C and D. $n(x) = \frac{1}{1+x}$, $v(x) = x$, p's values in $\mathbb{R}^2$, $p(A) = [0, 0]$, $p(B) = [0, 2.5]$, $p(C) = [2.5, 0]$ and $p(D) = [2.5, 2.5]$ (a square).

In this example, having all agents in a single group, $\mathcal{G}$, is the best solution. $\forall_{a\in\{A,B,C,D\}} U_p(a, \mathcal{G}) = \frac{4}{1+2.5\sqrt{2}/2} > 1$. But this situation could not be reached from groups consisting of a single agent by individual rational decisions, since the best interest one single agent could get from grouping with another agent would be $\frac{2}{2.25} < 1$, so no agent would want to group with any other. With our algorithm, if we start with a group $\mathcal{G} = \{A\}$, due to making computations based on the potential maximum size of the group rather than its actual size, individual interest of B and C for join the group would be estimated at $\frac{4}{3.5} > 1$ with an atomic variant (even more with a non-atomic variant) and the group $\mathcal{G}$ will be updated $\mathcal{G} = \{A, B, C\}$. On the second run of the inner loop, with the bary center of $\mathcal{G}$ being at $[2.5/3, 2.5/3]$, the atomic-estimated interest for D to join $\mathcal{G}$ would be $\frac{4}{1+5/3\sqrt{2}} > 1$. We end up with $\mathcal{G} = \{A, B, C, D\}$.

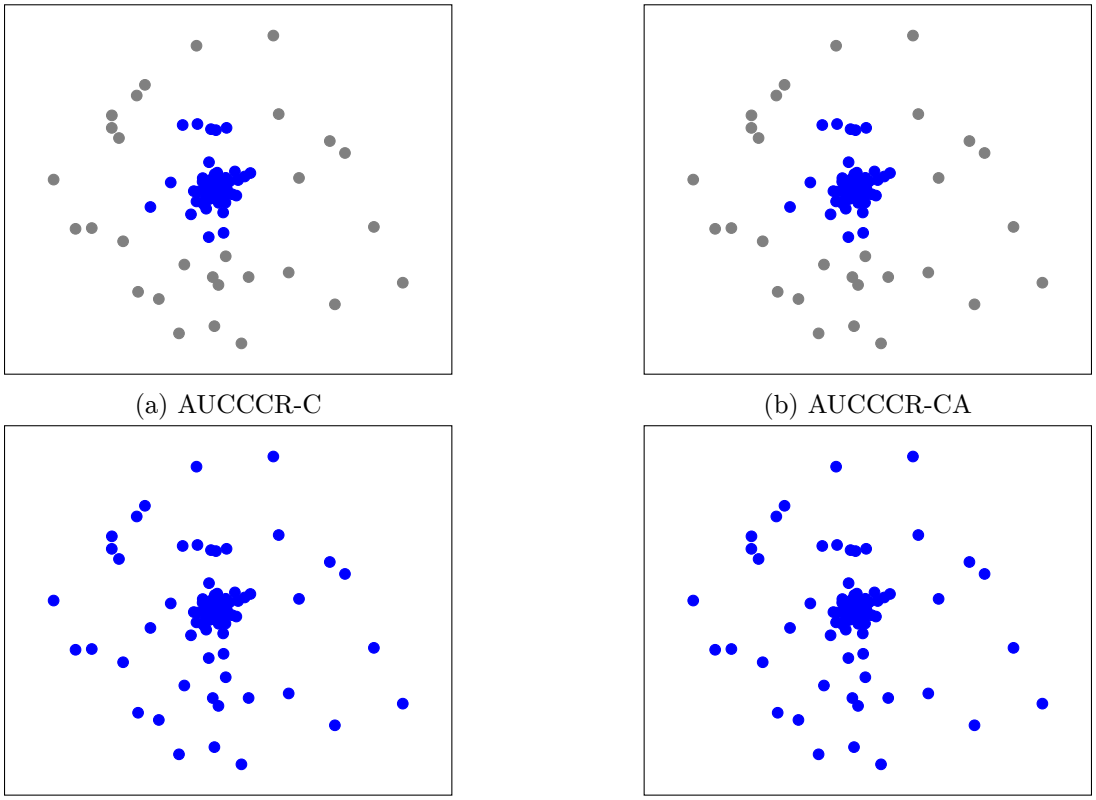
IV Experimentation

In this section, we evaluate Algorithm 3, in normal an atomic agents variants and see how much, in practice, it deviates from Algorithm 2's Nash equilibrium property. We compare our algorithm with two reference clustering algorithms: OPTICS [Ank+99] and k-means [Llo82] with k-means++ [AV07] initialization. All algorithms evaluated have guaranteed termination. We perform this evaluation on four kinds of datasets: synthetic data and three real data applications: leisure travel, buying and machine-learning.[1]

IV.1 Synthetic Data

We first evaluate our algorithm on $\mathbb{R}^2$ data generated using a combination of Gaussian distributions. We choose Gaussians because those distributions are usually considered good models of characteristics distribution in real populations. A fixed number of points is generated, the distribution used to generate each vector is chosen randomly according to a predetermined ratio. Parameters of these laws will be detailed in each specific tests. We used the usual euclidean distance for d, $n(x) = \frac{1}{1+x}$ and

[1] Our code is available at https://gitlab.inria.fr/abouchra/distributed_neural_networks



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

Figure 1: Clusters found in Bi-Gaussian

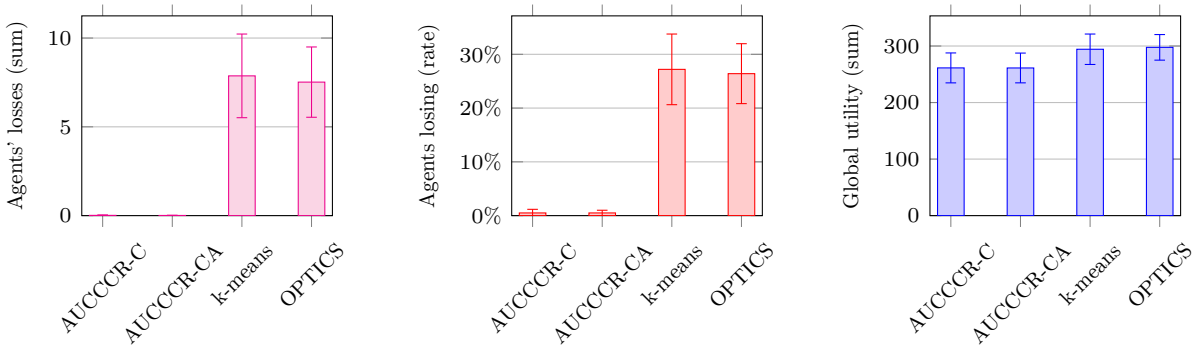
$v(x) = \sqrt{x}$. $prc = 20$ (number of trials with the same parameters), $mmt = 5$ (momentum). Results are average over 10 runs, error bars indicate standard deviation. Example results, based on a single run (same data for all algorithms), are also given. In these, a color identify a cluster, gray points are alone (or in a cluster of size 1). "AUCCCR-C" is Algorithm 3 in its normal variant, "AUCCCR-CA" is the atomic variant.

IV.1.a Bi-Gaussian

For this test, we used two $(0, 0)$-centered Gaussian distributions with 1 and 8 as standard deviations and a $0.5 - 0.5$ ratio (same probability for each distribution). The number of point is 100.

Figure 1 presents a sample output of the four algorithms. In these, a color identify a cluster, gray points are alone (or in a cluster of size 1). Figure 2c shows the global utility, as a sum of all agents' utility, for each algorithm. Figure 2a shows the sum of the losses of agents, which is the difference between the utility an agent gets from the present cluster affectation and the maximum utility the agent could get if the agent chooses its cluster (or being alone) following its own interest. Figure 2b shows the share of agents experiencing losses.

In this test, we see that AUCCCR-C(A) yields a lower global utility but also that its output is very close to a Nash equilibrium (near 0 losses), while k-means and OPTICS have more than 25% of agents experiencing losses. Looking at individual runs, we see that k-mean and OPTICS just produce a single giant cluster, maximizing its value for all agent but causing losses for agents that lie far away from the barycenter, while our algorithm, because it allows agent to stay on their own, took more care of their individual interest.



(a) Agents' losses (b) Share of agents having losses (c) Global utility achieved

Figure 2: Metrics in Bi-Gaussian for all algorithms

IV.1.b 3 Gaussians

For this test, we used three Gaussian distributions, $(-6, 0)$, $(0, 0)$ and $(0, 6)$-centered with 1.5 as standard deviations and a $0.33-0.33-0.33$ ratio (same probability for each distribution). The number of point is 200.

Figure 3 presents a sample output of the four algorithms. In these, a color identify a cluster, gray points are alone (or in a cluster of size 1). Figure 4c shows the global utility, as a sum of all agents' utility, for each algorithm. Figure 4a shows the sum of the losses of agents, which is the difference between the utility an agent gets from the present cluster affectation and the maximum utility the agent could get if the agent chooses its cluster (or being alone) following its own interest. Figure 4b shows the share of agents experiencing losses.

In this test, we see that AUCCCR-C(A) and k-means both perform well it terms of global utility, better than OPTICS, likely due to OPTICS density-centered design being ineffective for this kind of pattern ; OPTICS has difficulties in distinguishing close but clearly distinct distributions. Losses were limited but AUCCCR clearly outperformed k-mean and, even more, OPTICS.

IV.1.c Gaussians Star

For this test, we used five Gaussian distributions, $(0, 0)$, $(-10, -10)$, $(-10, 10)$, $(10, -10)$ and $(10, 10)$-centered with 5 for the $(0, 0)$ and 2 for others as standard deviations and a $0.5 - 0.125 - 0.125 - 0.125 - 0.125$ ratio. The number of point is 300.

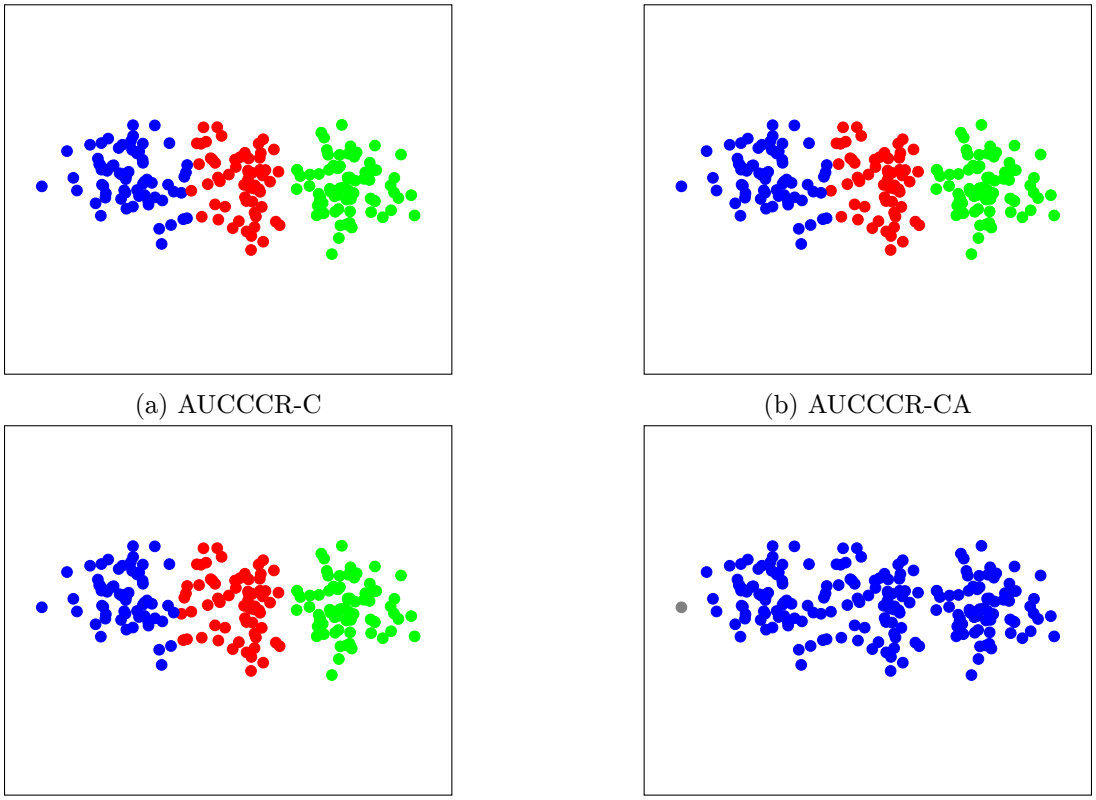
Figure 5 presents a sample output of the four algorithms. In these, a color identify a cluster, gray points are alone (or in a cluster of size 1). Figure 6c shows the global utility, as a sum of all agents' utility, for each algorithm. Figure 6a shows the sum of the losses of agents, which is the difference between the utility an agent gets from the present cluster affectation and the maximum utility the agent could get if the agent chose its cluster (or being alone) following its own interest. Figure 6b shows the share of agents experiencing losses.

In this test, all algorithms gave similar results in terms of global utility (slightly lower for AUCCCR on average). The gap much higher on losses, with AUCCCR clearly leading. Looking at samples, it seems that AUCCCR performed pretty well for identifying the five Gaussians mixed in the input data compared to k-means and OPTICS.

IV.1.d Effect of the number of agents

In this test we reused our Gaussians Star setup, but we tested different numbers of agents.

Figure 7c shows the global utility, as an average of all agents' utility, for each algorithm. Figure 7a shows the average of the losses of agents, which is the difference between the utility an agent gets from the present cluster affectation and the maximum utility the agent could get if the agent chooses its cluster (or being alone) following its own interest. Figure 7b shows the share of agents experiencing



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

Figure 3: Clusters found in 3 Gaussians

losses.

We see here that all algorithms have similar performances in terms of global utility; AUCCCR being slightly better for a high number of agents. In terms of losses, however, OPTICS and k-means seems to perform better for low numbers of agents. This is likely due to the fact that AUCCCR is used with an algorithm to find optimal k which optimizes global utility rather than loss. For k-mean and OPTICS, both simply put all agents in one cluster, which, for this number of agents, is a Nash equilibrium, but not necessarily an optimal affectation. Used with this k researcher, AUCCCR ended up being better for global utility and not for losses, while it is designed to minimize losses, in this specific case were a trivial (one cluster) affectation is a Nash equilibrium. Note that the losses are still very low compared to what k-means and OPTICS did for lower numbers of agents.

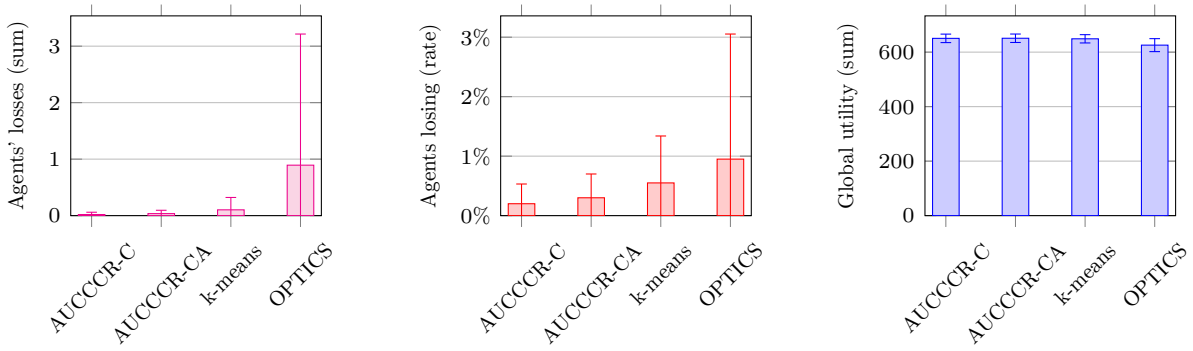
IV.1.e Effect of the standard deviations

In this test we reused our Gaussians Star setup, but we tested different standard deviations.

Figure 8c shows the global utility, as an average of all agents' utility, for each algorithm. Figure 8a shows the average of the losses of agents, which is the difference between the utility an agent gets from the present cluster affectation and the maximum utility the agent could get if the agent chooses its cluster (or being alone) following its own interest. Figure 8b shows the share of agents experiencing losses.

The value used in the "Standard deviations" axis is the $(0, 0)$-centered distribution's standard deviations. Others were modified by the same factor, as well as the distances between distributions' centers.

With a high standard deviation, the optimal solution becomes essentially to have nearly as many clusters as agents. k-means performs much worse than other algorithms (nearly all agents have losses, while nearly none have for other algorithms) because it is not designed to find this kind of affectation.



(a) Agents' losses (b) Share of agents having losses (c) Global utility achieved

Figure 4: Metrics in 3 Gaussians for all algorithms

AUCCCR and OPTICS have similar performance overall, except for 10. In losses, AUCCCR is the only algorithm to always remain at very low values. For the value 10, while AUCCCR was slightly lower for global utility, it is by far the best for losses.

IV.1.f Effect of scale

In this test we reused our Gaussians Star setup and combined the two previous tests, modifying at the same time, standard deviations, number of agents and mean values separations by the same scale factor (for geometrical reasons, this implies that the distances were increased by the square root of this factor).

Figure 9c shows the global utility, as an average of all agents' utility, for each algorithm. Figure 9a shows the average of the losses of agents, which is the difference between the utility an agent gets from the present cluster affectation and the maximum utility the agent could get if the agent chooses its cluster (or being alone) following its own interest. Figure 9b shows the share of agents experiencing losses.

Here we observe that AUCCCR can have difficulties with certain scale values, around 2 or 4, in terms of global utility, while being more competitive for a higher value of 8. This is probably due to such scales offering more opportunity for AUCCCR to deviate from the global optimum in favor of reducing individual losses. In terms of losses, however, AUCCCR remains extremely low for all value, while other algorithms get much worse results.

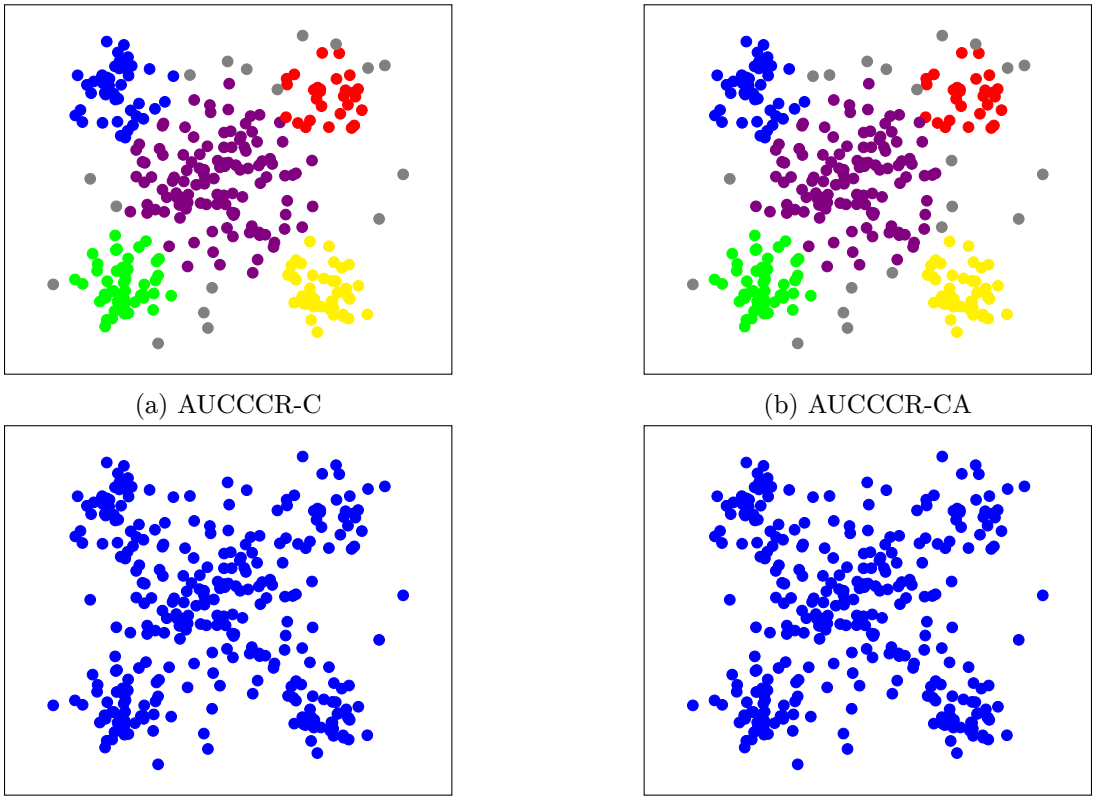
IV.1.g Size constraint

To evaluate the ability of our algorithm to manage cases where too large clusters are not wanted (agents want to be in a sufficiently large group but not too large) we take a simple case, with a single Gaussian, in which our usual $v(x) = \sqrt{x}$ function would make a single cluster (with all agents in it) optimal. We then change $v(x)$ so that too large clusters are not optimal.

We test two different functions, both $= \sqrt{x}$ for $x \leqslant 20$, but for $x > 20$ the first function is constant $\sqrt{20}$ while the second is decreasing $\sqrt{\frac{20}{1+\frac{|20-x|}{20}}}$ (both functions are continuous). The number 20 is an arbitrary choice and can be considered as a target size for clusters. See Figure 10 for a plot.

Figure 11 presents the clusters obtained with the different algorithms and $v(x)$ functions. You can see the precise size of each cluster in Figures 12 ($\sqrt{x}$), 13 ($\sqrt{20}$) and 14. ($\sqrt{\frac{20}{1+\frac{|20-x|}{20}}}$). Metric are shown in Figures 15 ($\sqrt{x}$), 16 ($\sqrt{20}$) and 17. ($\sqrt{\frac{20}{1+\frac{|20-x|}{20}}}$).

We see that changing the $v(x)$ function is a good way to adapt AUCCCR to size constraints. $\sqrt{20}$ is sufficient as the effect of distance reduces interest for larger (necessarily sparser) groups, even through $\sqrt{\frac{20}{1+\frac{|20-x|}{20}}}$ has a stronger effect. A downside of $\sqrt{\frac{20}{1+\frac{|20-x|}{20}}}$ is that AUCCCR is not designed for



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

Figure 5: Clusters found in Gaussians Star

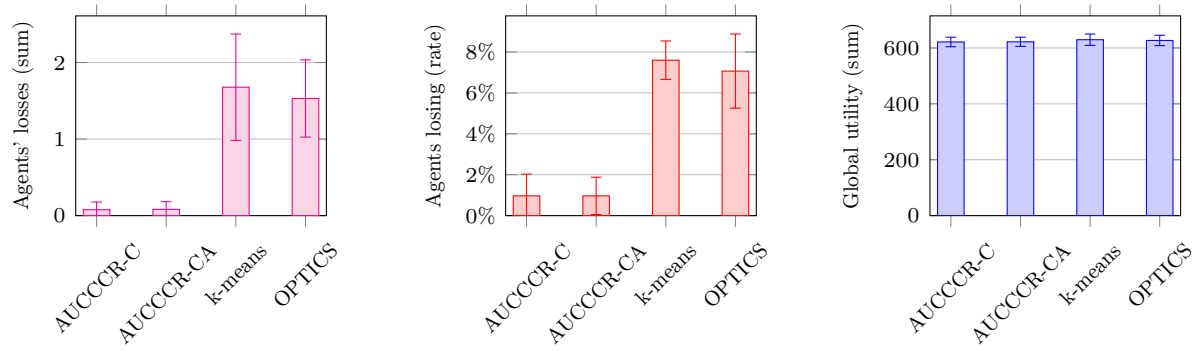
decreasing $v(x)$ functions and we see that we get (too) small clusters in that case, but this does not prevent the algorithm from working correctly. AUCCCR performs quite well in terms of size of cluster (close to the target size). AUCCCR clusters tends to be large, some times a bit larger than wanted, compared to k-means and, even more, OPTICS, which produce too small clusters. With the metrics, we see that our algorithm continues to outperform others in the same way as with other experiments with $\sqrt{20}$, with $\sqrt{\frac{20}{1+\frac{|20-x|}{20}}}$ however, k-means' results are very close to AUCCCR's.

IV.2 The BuddyMove dataset

The BuddyMove[RA14] dataset consists of statistics from users of a travel review website. For each user, the dataset gives the number of review wrote for each of 6 classes of destination (e.g. religious site, park, etc). From these data, we derived for each user the share of reviews written for each destination class. This can be interpreted as the relative interest of a user for each kind of destination. This could allow providing recommendation to users on whom they should go with for group travels, based on the similarity of their preferences.

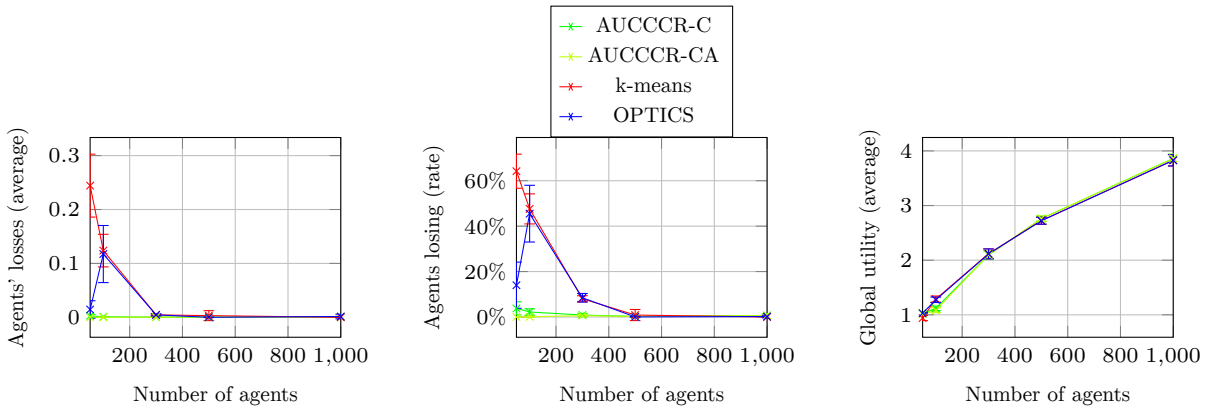
IV.2.a Hyperparameters

Our clustering algorithms used Euclidean distance for d, $n(x) = \frac{1}{1+x}$ and $v(x) = \sqrt{x}$. $prc = 20$, $mmt = 5$. Results are average over 10 runs, error bars indicates standard deviation. Sample examples, based on a single run, are also given. In these, a color identify a cluster, gray points are alone (or in a cluster of size 1). "AUCCCR-C" is Algorithm 3 in its normal variant, "AUCCCR-CA" is the atomic variant. An additional Hyperparameter is used: scale. The scale describes how "far" a given distance is considered by the algorithm (the distances naturally present in the dataset must be given an absolute cardial signification for the algorithm or its formalization make no sens). For example, if the scale is



(a) Agents' losses caused (b) Share of agents having losses (c) Global utility achieved

Figure 6: Metrics in Gaussians Star for all algorithms



(a) Agents' losses caused (b) Share of agents having losses (c) Global utility achieved

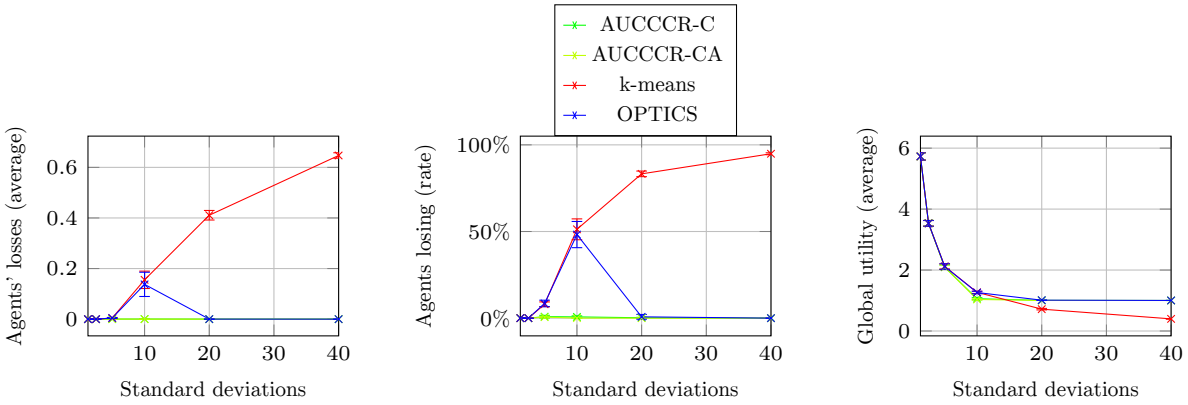
Figure 7: Metrics in Gaussians Star for different numbers of agents by different algorithms

10, a distance of 1 will be computed as 10 for computation. If the scale is bigger, distance between points will be considered longer by the algorithm.

IV.2.b Metrics

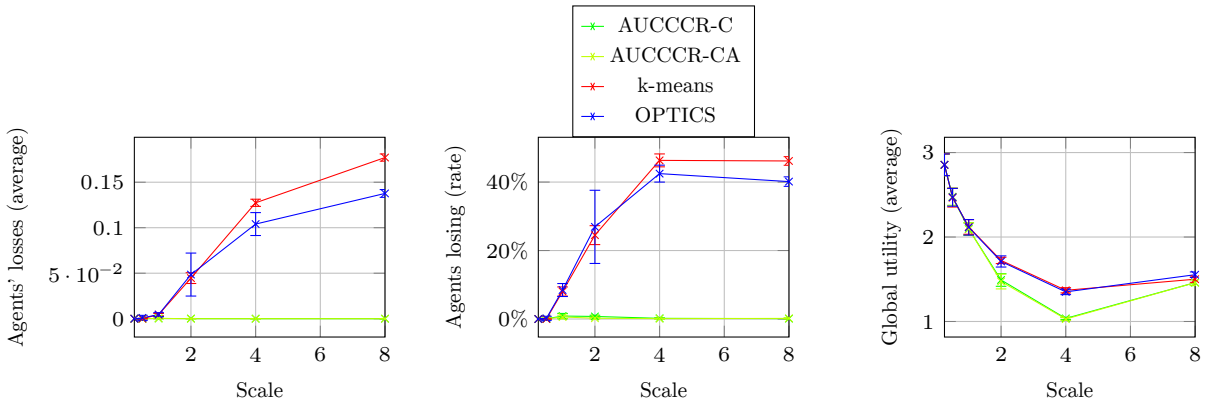
The graphs in Figure 18 presents the 3 following metrics : the average utility of agents (Figure 18c), the average losses of agents (Figure 18a) and the share of agents having losses (Figure 18b) for a range of scale values. The range of scale is selected to cover all the range of cases from a situation where everyone alone is optimal to a situation were a universal cluster is optimal.

We note that, is those graphs, the lines for the two variants of AUCCCR are largely superposed, and k-means and OPTICS are identical. For a low scale, users are considered close and all algorithms will detect a single cluster but, as the scale increases, the results become very different. AUCCCR, contrary to the reference algorithms, will prevent individual losses ; as we can see, losses are very low for AUCCCR, especially for large scales, while, for other algorithms, losses increase dramatically. For a scale of 130, AUCCCR has nearly 0 agents losses while k-mean and OPTICS have 30%. This comes at the cost of a reduced global utility, which slowly drops until it nearly hits 1 (equivalent to everyone alone) at 130. For midrange values, we see that AUCCCR has nearly no losses with a still high global utility (1.5, 1.7 for reference algorithms; this lower value being due to the small size of clusters possible with minimal individual losses) while reference algorithms have 15% to 20% of losses.



(a) Agents' losses caused (b) Share of agents having losses (c) Global utility achieved

Figure 8: Metrics in Gaussians Star for different standard deviations by different algorithms



(a) Agents' losses caused (b) Share of agents having losses (c) Global utility achieved

Figure 9: Metrics in Gaussians Star for different scales by different algorithms

IV.2.c Clusters size

The graphs in Figure 19 gives the obtained clusters sizes for the different algorithms with a scale of 115. We see that, for this midrange scale, k-means and OPTICS identified only a universal cluster, while both AUCCCR variants identified 3 and let numerous users alone.

IV.2.d Clusters visualisation

Since the dataset has a large dimension (6), we provide views in reduced dimensions, 2 and 3. We propose two different kinds of dimension reduction methods. The first simply presents the dimensions were the original dataset has higher standard deviation. See Figure 20 for 2D and Figure 21 for 3D. The second uses the *scikit-learn*[Ped+11] Python library, which implements various algorithms that can perform non-linear dimension reduction. The specific algorithm we used is MDS (Multi-Dimensional Scaling)[BG97]. See Figure 22 for 2D and Figure 23 for 3D.

IV.3 The Wholesale dataset

The Wholesale [Fer11] dataset consists of statistics from customers of a wholesale vendor. For each customer, the dataset indicates the annual spending for each of 6 classes of products (e.g. fresh, frozen, etc). From these data, we derive for each customer the share of reviews written for each product class. This result could for instance be used to provide recommendations to customers on whom they should collaborate with to make grouped orders more directly, removing the wholesale distributor from the

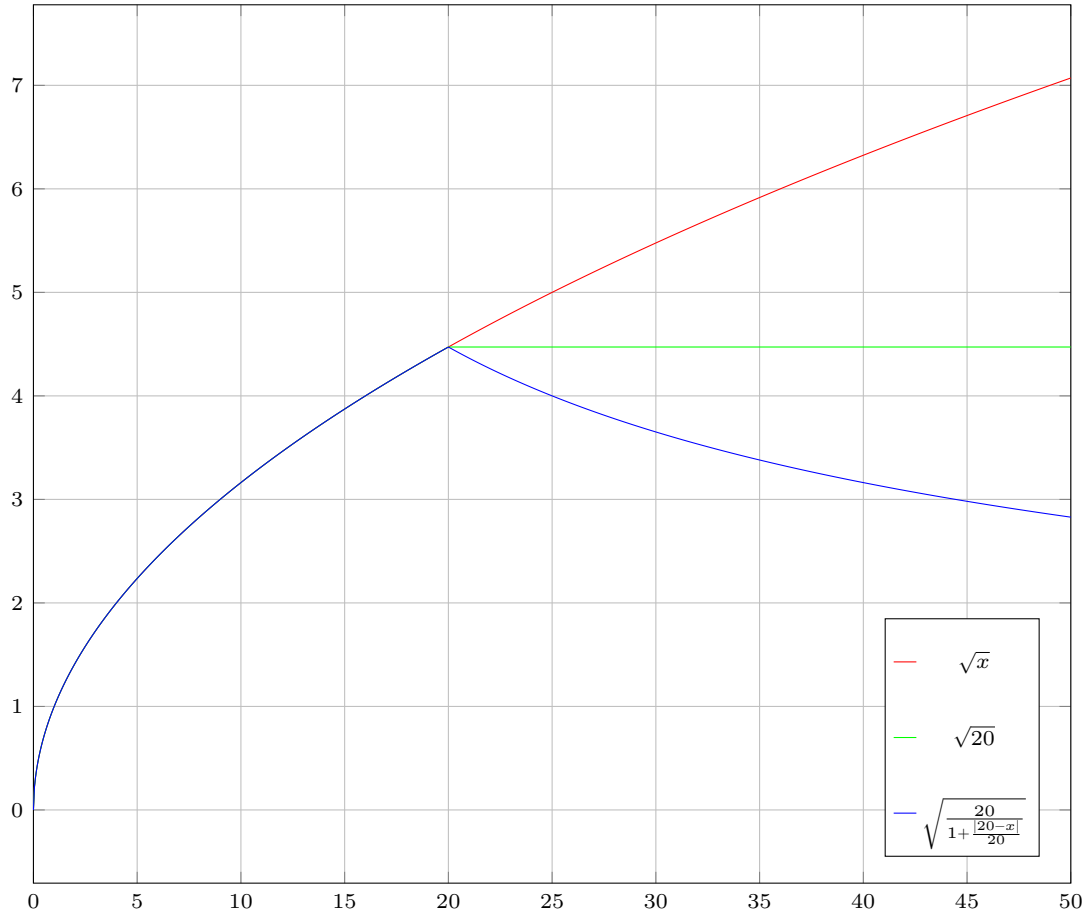


Figure 10: Group value functions

circuit (short circuit distribution), based on which kind of products they usually order.

IV.3.a Hyperparameters

We configure our clustering algorithms to use the Euclidean distance for d, $n(x) = \frac{1}{1+x}$ and $v(x) = \sqrt{x}$. $prc = 20$, $mmt = 5$. Results are average over 10 runs, error bars indicate standard deviation. Sample examples, based on a single run, are also given. In these, a color identify a cluster, gray points are alone (or in a cluster of size 1). "AUCCCR-C" is Algorithm 3 in its normal variant, 'AUCCCR-CA" is the atomic variant. An additional Hyperparameter is used: scale. The scale describes how "far" a given distance is considered by the algorithm (the distances naturally present in the dataset must be given an absolute cardial signification for the algorithm or its formalization make no sens). For example, if the scale is 10, a distance of 1 will be computed as 10 for computation. If the scale is bigger, distance between point will be considered longer by the algorithm.

IV.3.b Metrics

The graphs in Figure 24 presents the 3 following metrics : the average utility of agents (Figure 24c), the average losses of agents (Figure 24a) and the share of agents having losses (Figure 24b) for a range of scale values. The range of scale is selected to cover all the range of cases from a situation where everyone alone is optimal to a situation were a universal cluster is optimal.

We note that, is those graphs, the lines for the two variants of AUCCCR are largely superposed. For a low scale, users are considered close and all algorithms will detect a single cluster but, as the scale increases, the results become very different. AUCCCR, contrary to the reference algorithms, will prevent individual losses ; as we can see, losses are close to 0 for AUCCCR, especially for large scales,

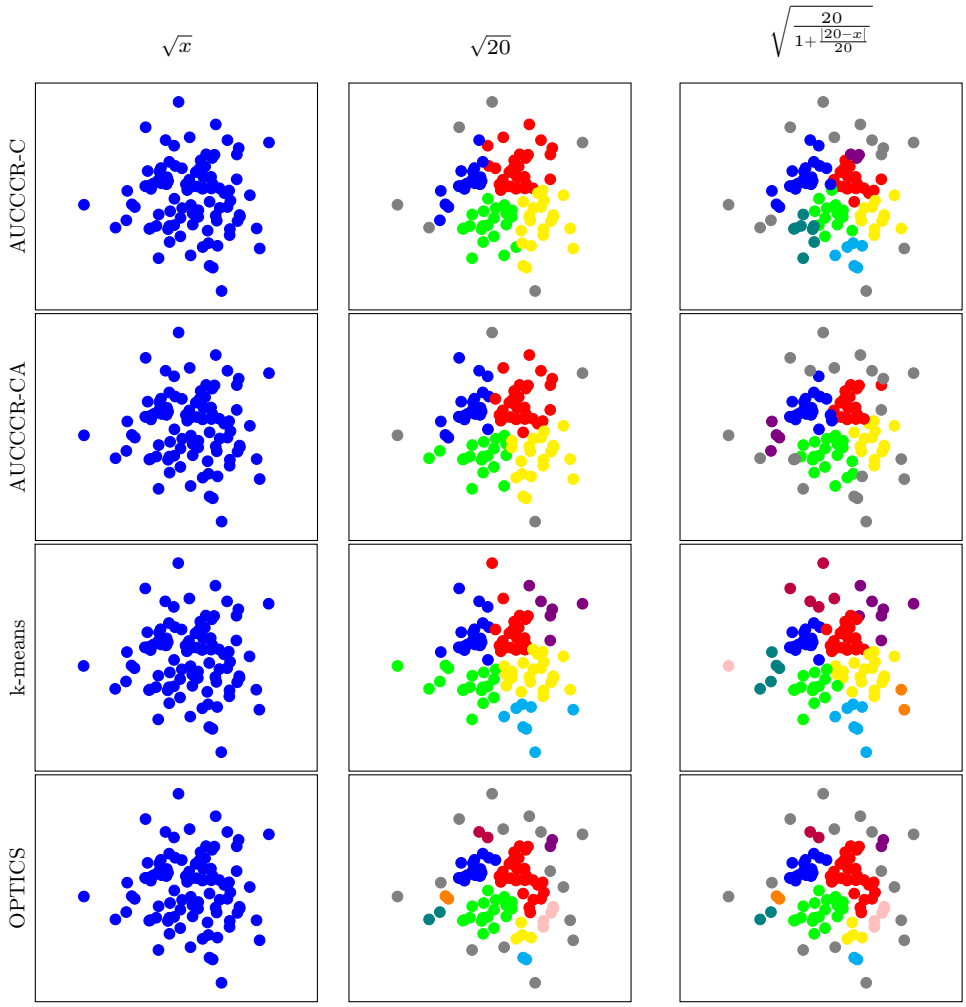


Figure 11: Clusters with various algorithms and different $v(x)$

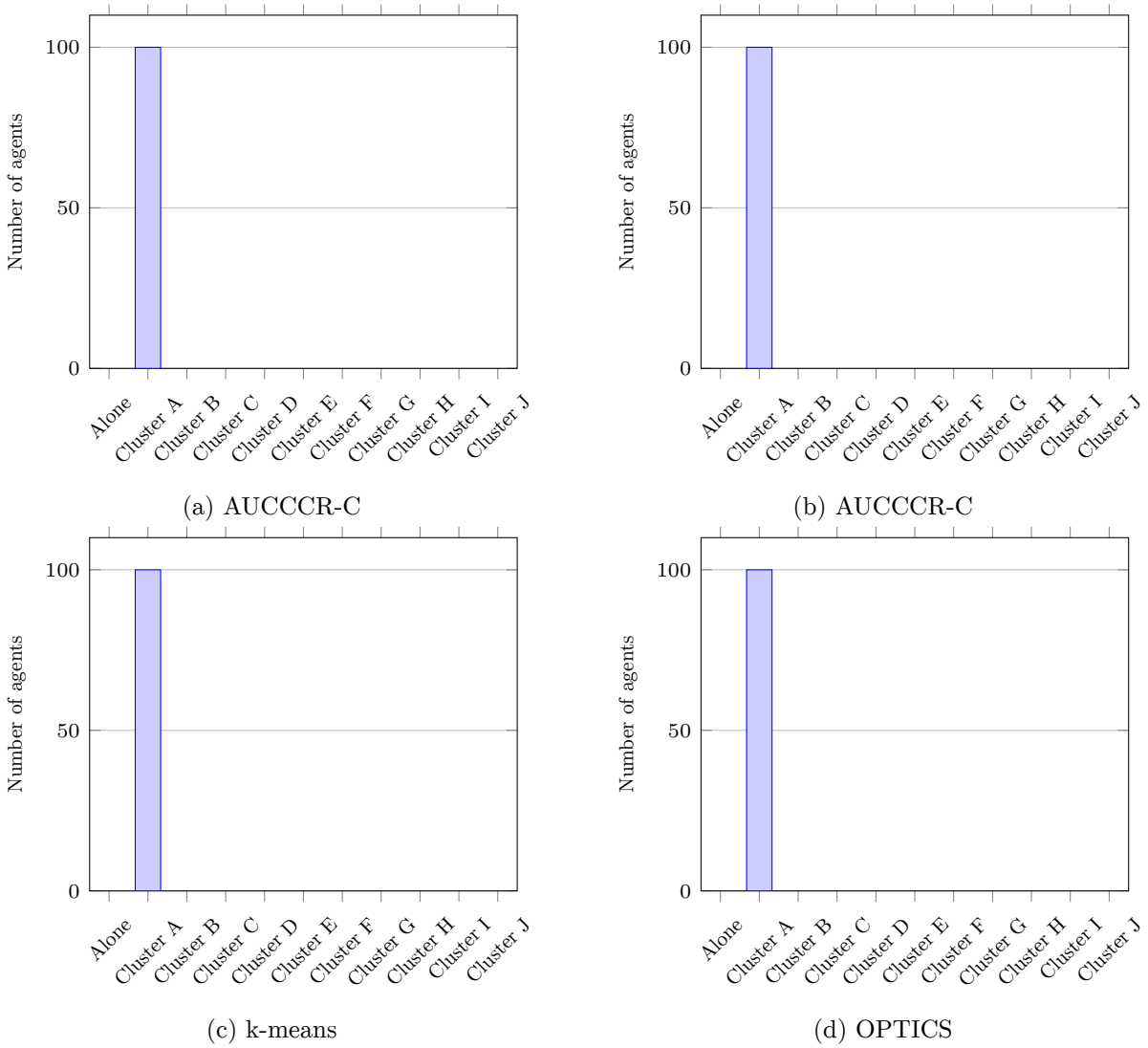
while, for other algorithms, losses increase dramatically. For a scale of 60, AUCCCR has nearly 0 agents losses while k-mean has $> 30\%$ and OPTICS has $> 40\%$. The global utility remains similar to k-means and greater than OPTICS for the lower half of scales, only higher values have a utility lower than OPTICS but still very close from both, OPTICS and k-means. For midrange values, we see that AUCCCR has nearly no losses with a global utility close to k-means and higher than OPTICS, while reference algorithms have around 10% of losses.

IV.3.c Clusters size

The graphs in Figure 25 gives the obtained clusters sizes for the different algorithms with a scale of 45. We see that, for this midrange scale, k-means and OPTICS identified only a universal cluster, while both AUCCCR variants identified 3 and let numerous users alone.

IV.3.d Clusters visualisation

Since the dataset has a large dimension (6), we provide views in reduced dimensions, 2 and 3. We propose two different kinds of dimension reduction methods. The first simply presents the dimensions were the original dataset has higher standard deviation. See Figure 26 for 2D and Figure 27 for 3D. The second uses the *scikit-learn*[Ped+11] Python library, which implements various algorithms that can perform non-linear dimension reduction. The specific algorithm we used is MDS (Multi-Dimensional Scaling)[BG97]. See Figure 28 for 2D and Figure 29 for 3D.



(a) AUCCCR-C

(b) AUCCCR-C

(c) k-means

(d) OPTICS

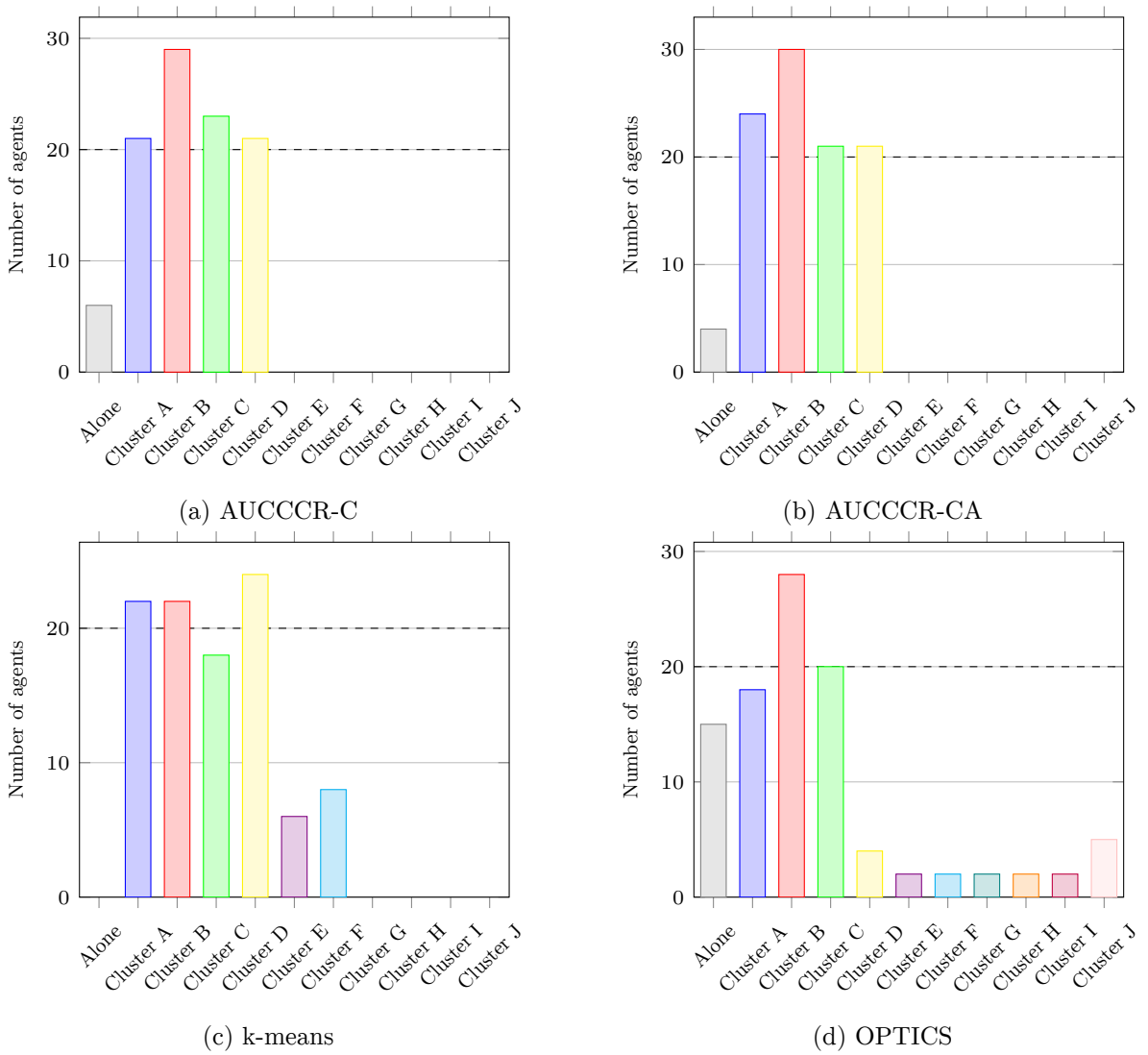
Figure 12: Clusters size with $\sqrt{x}$ by different algorithms

IV.4 Machine-learning Data - MNIST

As a more concrete application, we evaluate our program on machine-learning data. We use here classical multi-layer perceptrons (MLP) on the well-known MNIST dataset.

The task associated with the MNIST dataset is handwritten digits recognition. From this task, we want to obtain a cooperation recommendation task. There are various distributed machine-leaning methods, as the one presented in [BFT21]. These methods imply cooperation between different MLPs. Now, we want our MLPs to have more or less similar tasks. To this aim, we simply modified the MNIST task for some of those MLPs: part of our MLPs have to perform the standard digit recognition task, while others have to perform a modified task where two digits are exchanged, so a [9] would be classified as a eight and vis versa. The clustering task would be to assign to the same cluster MLPs with the same task and different clusters MLPs with different tasks.

Since our clustering algorithms takes real vectors as inputs, we need to make real vectors from our MLPs. The most straightforward way would be to directly take the MLPs learned parameters, which are real values, but this is not a good way. Due to the nature of neural networks, similar learned tasks absolutely do not imply similar learned parameters. For this reason, we used a different method. After a reasonable amount of training, we made each MLP work on the test part of the MNIST dataset. For each different digit tested (from 0 to 9), we took the average of the outputs of each MLP (10 real



(a) AUCCCR-C

(b) AUCCCR-CA

(c) k-means

(d) OPTICS

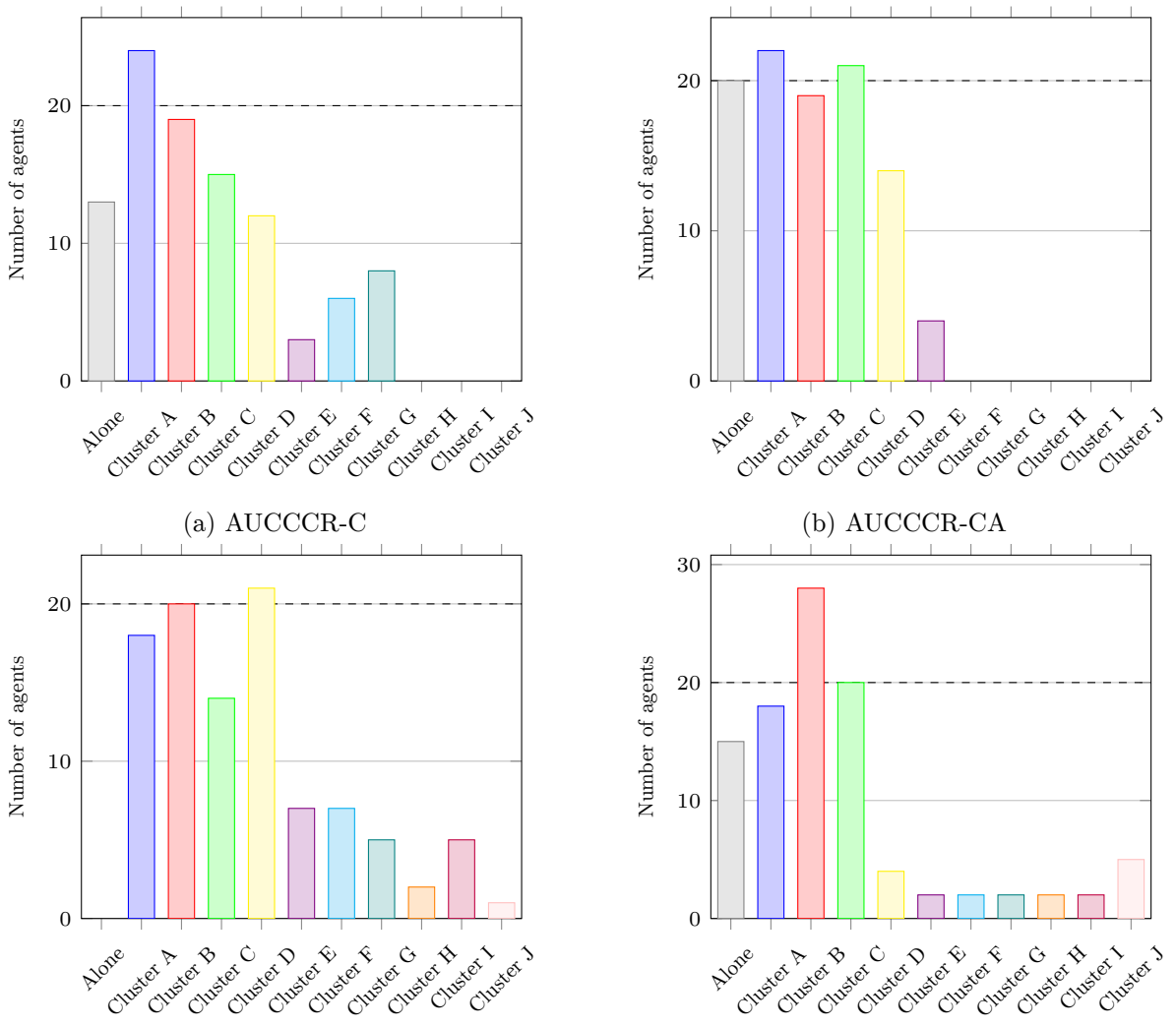
Figure 13: Clusters size with $\sqrt{20}$ by different algorithms

values, which are the MLPs estimated likeliness of the input image to be each of the digits from 0 to 9) and concatenated them in one $\mathbb{R}^{100}$ vector. These vectors will be used as input for our clustering algorithms.

We use two metrics to evaluate the quality of the outputted affectation: the Identification rate, which is the share of the pairs of MLPs with identical task being in the same cluster, and the Differentiation rate, which is the share of the pairs of MLPs with different tasks being in different clusters. We want both value to be as close to 1 as possible.

We trained 16 MLPs, 9 of them performing a standard classification task and 7 of them have 8 and 9 permuted. They were trained each on a different part of the MNIST dataset training part of size 3500. The MLPs had a 784=300=100=10 configuration with $\lambda = 1$ sigmoid activation functions and a 0.1 learning rate. The $\mathbb{R}^{100}$ vectors where built from the 1000 first values of the MNIST dataset test part.

Our clustering algorithms used euclidean distance for d, $n(x) = \frac{1}{1+15x}$ and $v(x) = \sqrt{x}$. $prc = 20$, $mmt = 5$. Results are average over 10 runs, error bars indicates standard deviation. Sample examples, based on a single run, are also given. In these, a color identify a cluster, gray points are alone (or in a cluster of size 1). "AUCCCR-C" is Algorithm 3 in its normal variant, 'AUCCCR-CA" is the atomic variant.



(a) AUCCCR-C

(b) AUCCCR-CA

(c) k-means

(d) OPTICS

Figure 14: Clusters size with $\sqrt{\frac{20}{1+\frac{|20-x|}{20}}}$ by different algorithms

The MLPs were trained with stochastic gradient descent, with a mini-batch size of 1000 and different number of mini-batches.

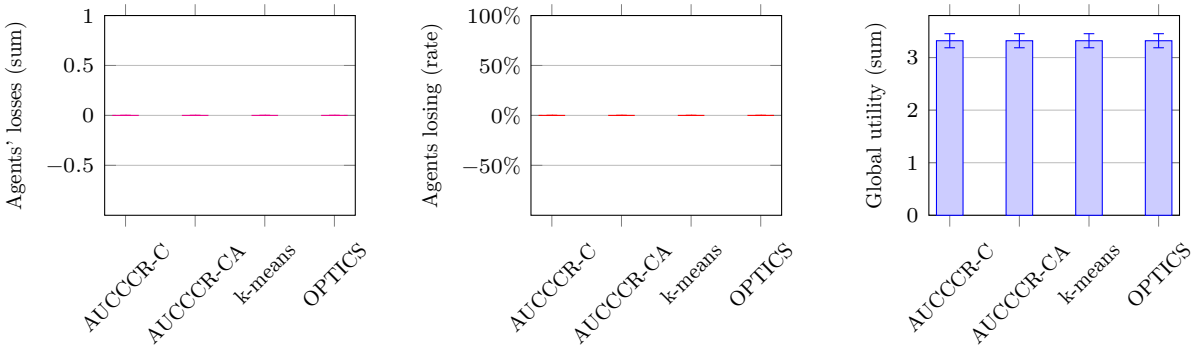
Figure 30a presents the identification rate for various numbers of mini-batches and Figure 30b the differentiation rate.

We essentially observe that all algorithms remains close to perfect for all tests. While those tests on a very simple case do not allow determining which algorithm would be the best, they prove the validity of the idea of using clustering algorithms to provide cooperation recommendations for distributed machine-learning.

Figure 31 gives examples of cluster affectation outputted by AUCCCR-C. Since we used vectors from a space of dimension 100, we selected to relevant dimensions, 89 and 99 (corresponding to permuted digits), to make these 2D plots.

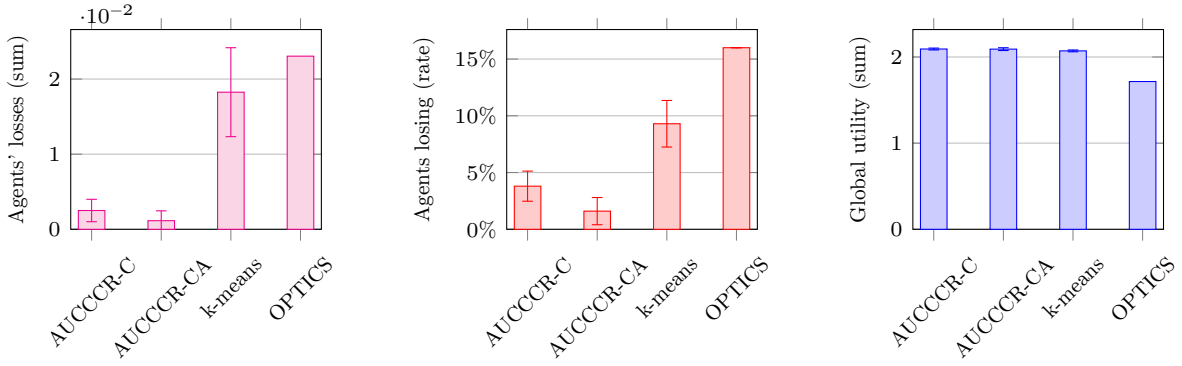
IV.5 Machine-learning Data - VSN

To go farer than our previous experiment with MNIST, we use a different task: vehicle recognition from sensors. We use a dataset from [DH04]. This dataset contains data from sensors (notably seismic and acoustic sensors) produced while some (military) vehicle passes near the sensor. In that case, no



(a) Agents' losses (b) Share of agents having losses (c) Global utility achieved

Figure 15: Metrics with $\sqrt{x}$ for all algorithms



(a) Agents' losses (b) Share of agents having losses (c) Global utility achieved

Figure 16: Metrics with $\sqrt{20}$ for all algorithms

obvious group affectation is possible without relying on clustering, allowing us to evaluate how our approach allows to effectively get better performance than a reference.

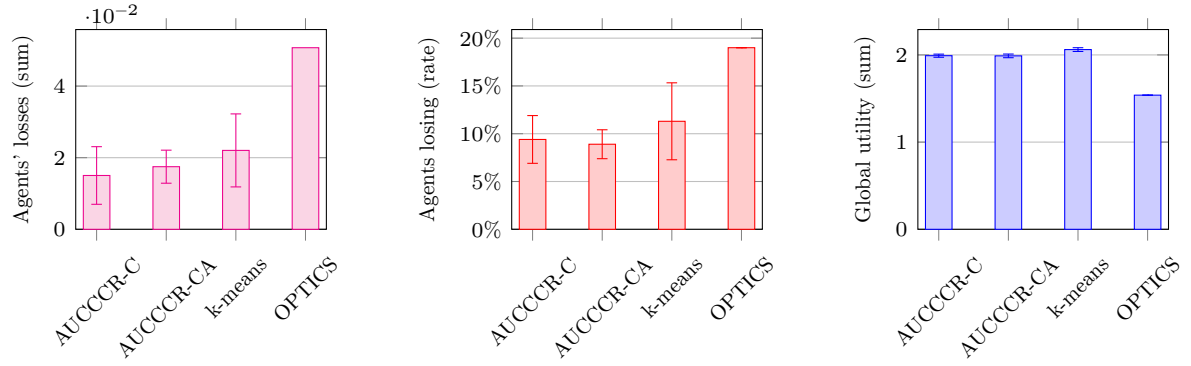
The task proposed is to recognize the class of vehicle (*assault amphibious* or *dragon wagon*) from each sensor's data (binary classification). To allow a MLP to perform this task, the data from each set of sensors (node) is first transformed into 50 seismic and 50 acoustic features (100 total inputs). This process (based on Fast Fourier Transform) is described in the original paper ([DH04]).

As for MNIST, we rely on the approach of distributed multi-task machine learning presented in [BFT21] For our distributed multi-task setup, we consider each node as a peer. The tasks are similar, since all peer what to classify the same kinds of vehicles from the same kind of data, but different due to the location of sensors influencing their output.

To get real vectors for our clustering algorithms, as we do with MNIST, we take locally pre-trained networks (one for each peer) and test them on a standard set inputs. From the obtained output, we construct $\mathbb{R}^2$ vectors from the average output of networks on each type of vehicle.

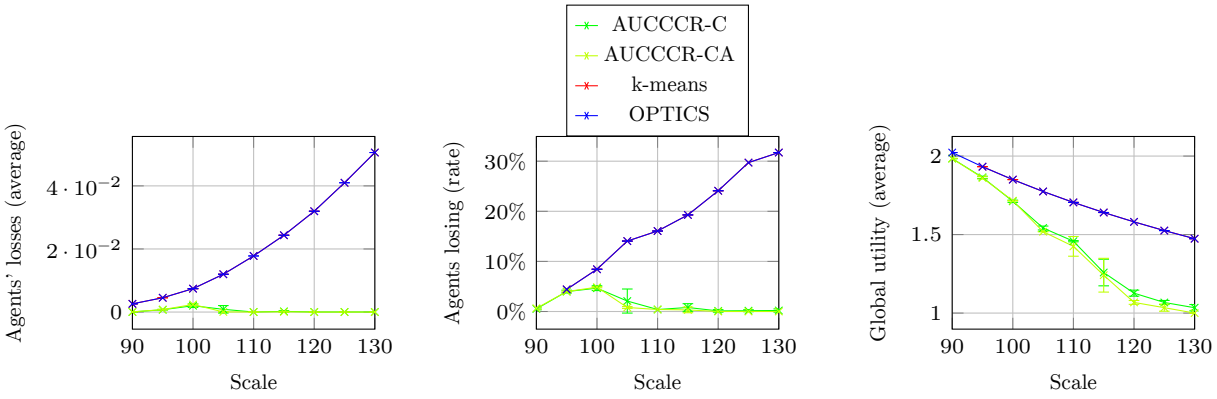
For this experiment we use 16 peers, each with a training set of 50 samples. The mini-batch size is 25. We use an MLP with a 100=50=20=1 layout, trained with stochastic gradient descent, $\lambda = 1$ sigmoid activation functions and a 0.1 learning rate.Our clustering algorithms used euclidean distance for d, $n(x) = \frac{1}{1+15x}$ and $v(x) = \sqrt{x}$. $prc = 20$, $mmt = 5$. Results are average over 10 runs.

We start with a local pre-training phase of 400 mini-batches. Then, we use the output of the pre-trained networks on a (identical for all peers) test set of 1000 samples to get inputs for clustering. With the output of our clustering algorithm, we define partial models (see [BFT21] for details) to use for the real training phase of 400 mini-batches (we restart from 0 for a fair comparaison with reference results) and perform tests on 200 samples (specific to each peer). Here, the performance



(a) Agents' losses (b) Share of agents having losses (c) Global utility achieved

Figure 17: Metrics with $\sqrt{\frac{20}{1+\frac{|20-x|}{20}}}$ for all algorithms



(a) Agents' losses caused (b) Share of agents having losses (c) Global utility achieved

Figure 18: Metrics in Buddymove for different scales by different algorithms

of our clustering algorithm is not evaluated by the affectation itself but by the performance of the resulting machine learning process (median accuracy).

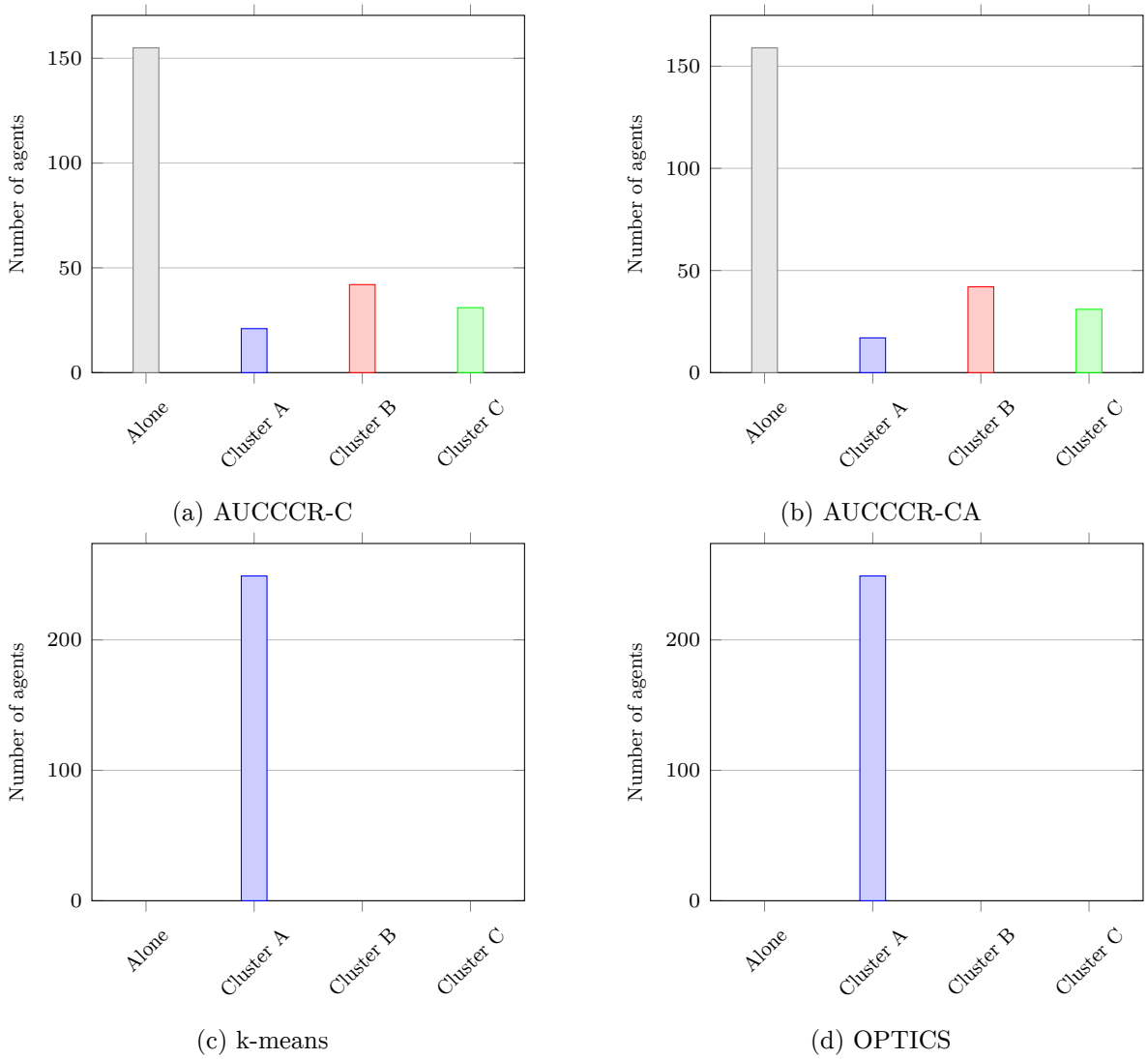
Figure 32 gives examples of cluster affectation outputted by different algorithms. Figure 33 compares the performance of the reference from [BFT21] to what can be achieved using affectations obtained from our clustering algorithms (k-means and OPTICS both failed to provide meaningful affectations).

While no obvious cluster affectation is visible, AUCCCR manages to find some clusters (differences between samples from the two variants of AUCCCR are more likely due to randomness than real difference between the two algorithms). k-means and OPTICS however both fail in this difficult case.

k-means and OPTICS fail to provide meaningful clustering information, not allowing any gain over reference (all logical affectations derivable from their outputs are in reference). With AUCCCR, however, we get two cluster affectations, from each of which can be derived several averaging schemes (see [BFT21] for details).

We present results of three different schemes, which can be considered the most obvious possibilities:

- **per class:** A 100-50-20-1 (full network) model for each cluster (out of clusters peers are trained locally).
- **semi-local:** A 100-35-14-1 for all peers and a 0-15-6-0 (dependent on the previous one) for each cluster (leaving nothing local for peers in clusters).
- **semi-local+local:** A 100-35-14-1 for all peers and a 0-7-3-0 (dependent on the previous one)



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

Figure 19: Clusters size in Buddymove for scale 115 by different algorithms

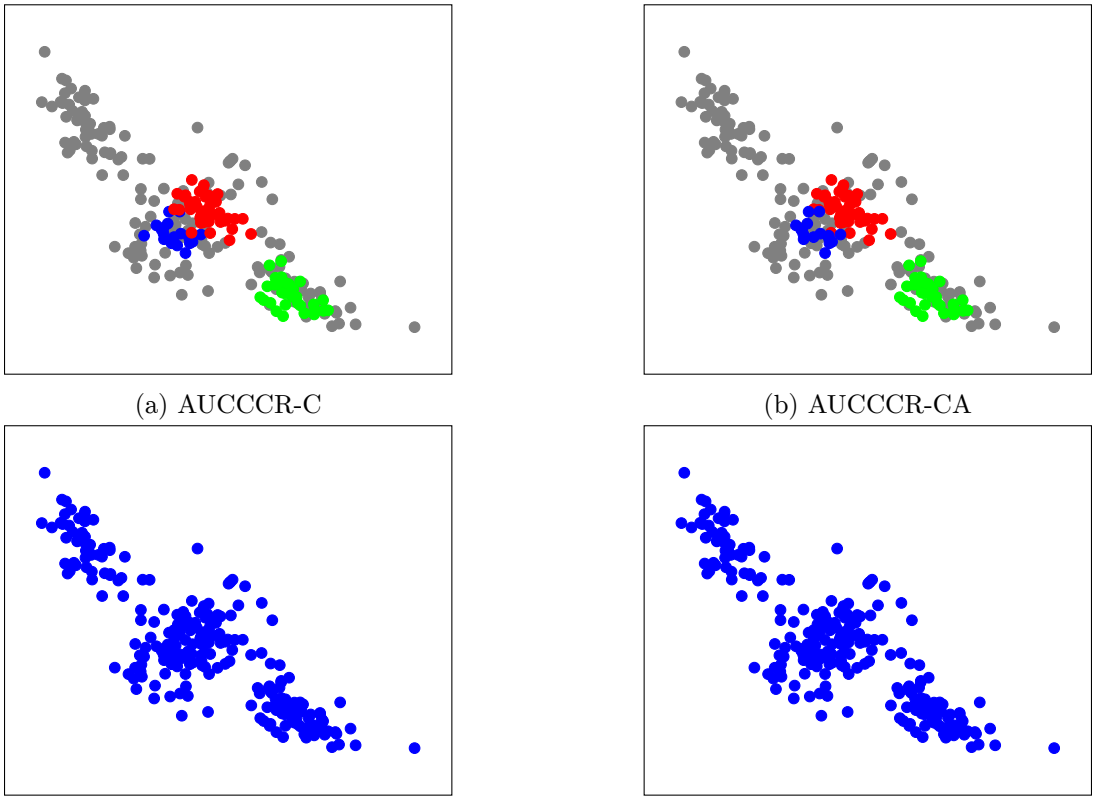
for each cluster (leaving 0-8-3-0 local for peers in clusters).

Notice that those are only there of the numerous possibilities offered in our case by [BFT21]'s approach. We chose those because they are the most consistent with results from [BFT21] but further optimization is possible and may lead to better results, even through [BFT21] concluded that the solution we used in semi-local+local was the best in their experiments.

As we see in results, per class does not beat reference, likely due to lots of peers being out of classes (the performance is slightly higher than local learning). Semi-local achieve reference-like performance, likely because the gain from clusters is cancelled by the loss from lack of intra-cluster differentiation. Semi-local+local, however, manages to obtain (slightly) better than reference results. The gain is limited but similar to what reference achieved with partial averaging compared to global.

IV.6 General analysis of results

On our synthetic data tests, we showed that our algorithm is more efficient for cluster identification than classical clustering algorithms. It also appeared that, as intended, our algorithm causes much lower individual losses to agent than other algorithms. While it is not possible to completely remove losses in the absence of a Nash (or similar) equilibrium for mathematical reasons, our algorithm



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

Figure 20: Clusters found in Buddymove@115

manages to keep such losses low, reducing the risk of agent unsatisfaction and collapse.

Our algorithm also exhibited good performance on the two real datasets, Buddymove and Wholesale, proving its applicability to real use cases.

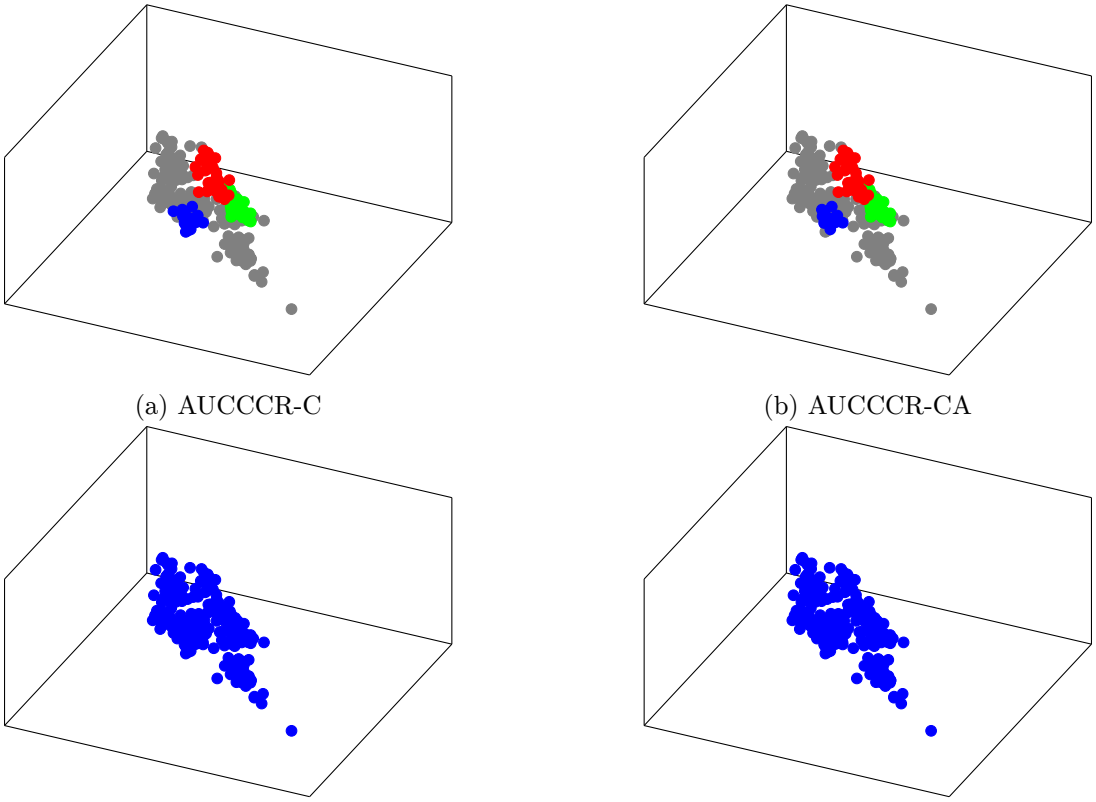
We also proved that our idea of using clustering algorithms to identify good collaboration opportunities in distributed machine learning is valid, with our MLP-MNIST example, and that in could lead to performance improvement in cases where no manual affectation is possible, with our MLP-VSN example.

V Related work

Hedonic games as a theoretical model were originally proposed in the economic community by [DG80]. More recently, this field has been studied by the algorithmic community. The authors of [AS16] summarize algorithmic studies of these games. Existing algorithmic solutions all relies on some kind of equilibrium (similar to Nash equilibrium) which may not exist in real applications.

In addition to those general researches on hedonic games, some researchers considered using hedonic games to solve practical problems. For example, [Saa+10] worked on collaboration between roadside units in Intelligent Transportation Systems. [Ang+18] considered hedonic games in the context of Fog Computing. Application of these games to edge computing is also proposed in [ZCY17]. These games can also be applied to energy networks, as suggested by [Mei+19]. Each of these articles presents its own model and algorithm, to solve the specific problem they are considering.

The most known clustering algorithm is k-means[Llo82]. A lot of variants have been proposed, notably k-means++[AV07], which improves its initialization. There are also variants of k-means with size constraints [GCT14; Tan+19] or density [DV05] but such variants can only find clusters with



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

Figure 21: Clusters found in Buddymove@115

similar characteristics (size/density) leading to results unwanted for our application (the graphs in [GCT14] show this clearly). Another well-known clustering algorithm is OPTICS[Ank+99], which can be seen as an improved DBSCAN[Est+96]; both OPTICS and DBSCAN are density-based (k-means is distance-based).

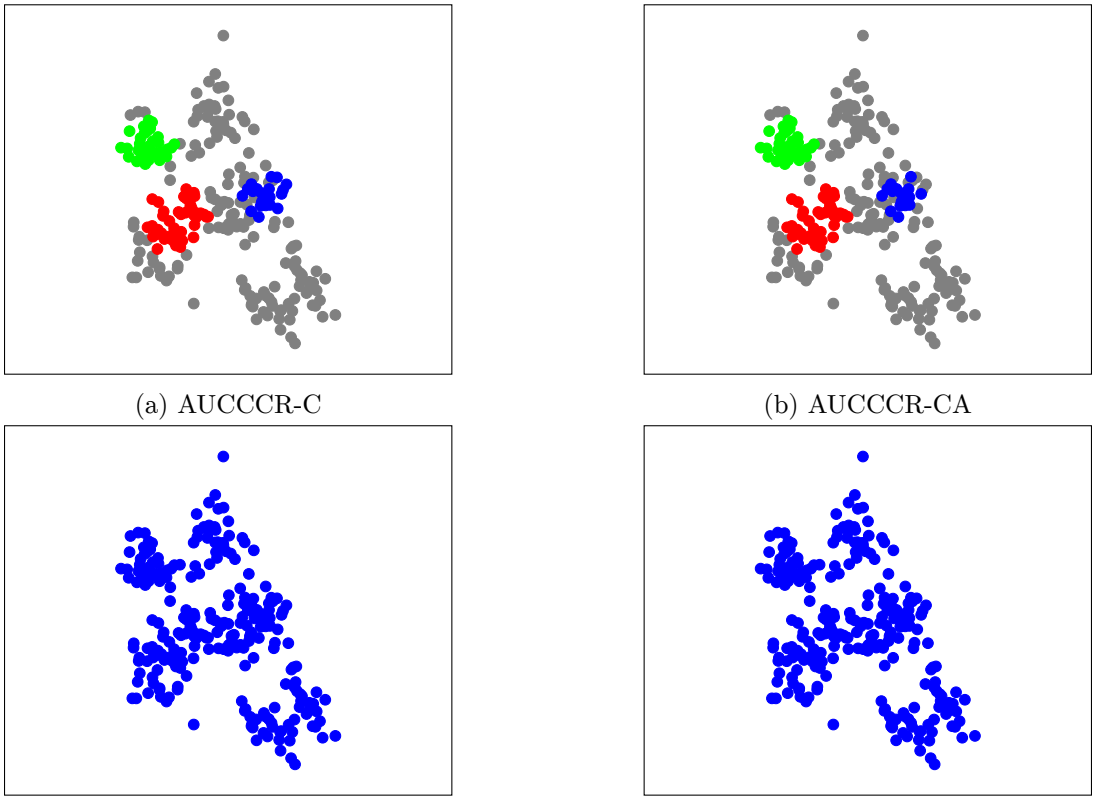
Through they did not formalized the problem as an hedonic game, the authors of [SvW19] also propose to use a clustering algorithm for coalition formation in the specific context of machine learning. This work did not proposed a novel algorithm and simply used a variant of DBSCAN. OPTICS, which we tested and proved limited in our experiments, being an improved DBSCAN, we can affirm that the solution proposed in [SvW19] would not exhibit good results compared to ours.

VI Conclusion

In this work, we defined a new class of hedonic games on which we based a algorithm for providing cooperation recommendation. We compared our algorithm to classical clustering algorithm that could have also been used for the same recommendation problem. We proved that our algorithm yields results that exhibit close to no losses and are thus much to a Nash equilibrium (which does not exist in the general case) than other algorithms.

References

[Ang+18] Cosimo Anglano et al. “A game-theoretic approach to coalition formation in fog provider federations.” In: *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2018, pp. 123–130.



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

Figure 22: Clusters found in Buddymove@115 (Manifold MDS)

[Ank+99] Mihael Ankerst et al. "OPTICS: Ordering Points To Identify the Clustering Structure." In: *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*. Vol. 28. 2. ACM Press, 1999, pp. 49–60.

[AS16] Haris Aziz and Rahul Savani. "Hedonic Games." In: *Handbook of Computational Social Choice*. Cambridge University Press, 2016, pp. 356–376.

[AV07] David Arthur and Sergei Vassilvitskii. "K-means++: The Advantages of Careful Seeding." In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Vol. 8. SODA '07. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

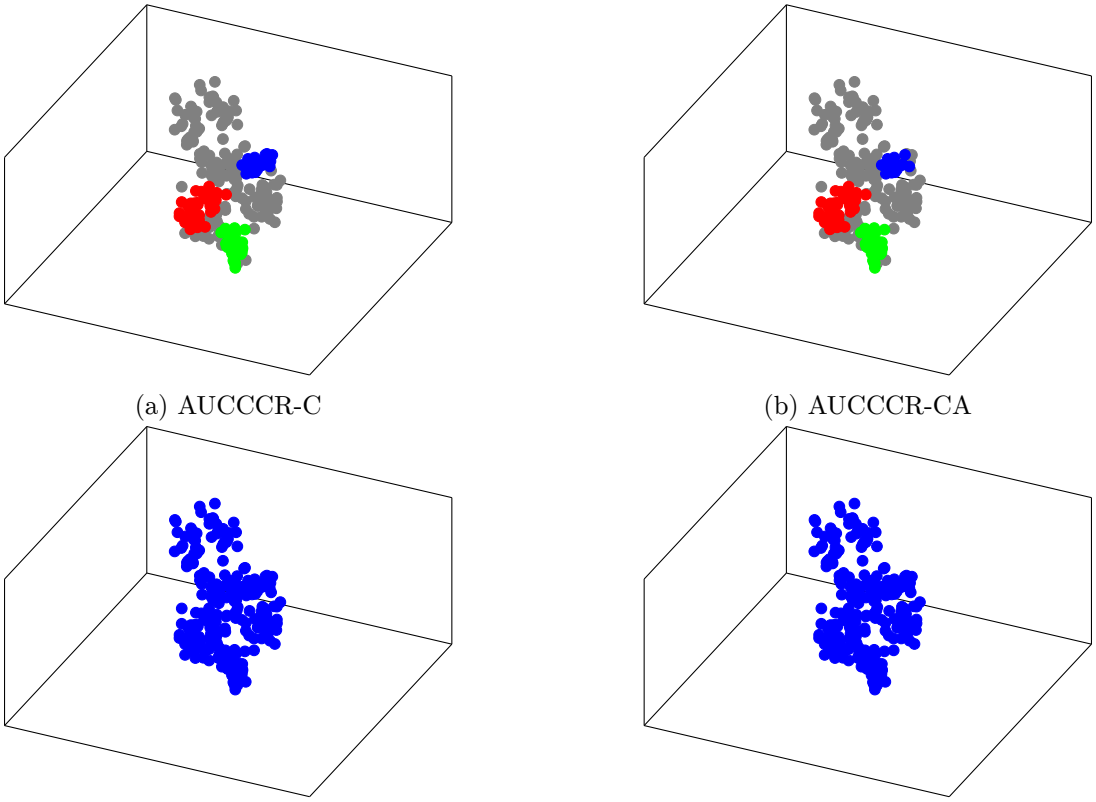
[BFT21] Amaury Bouchra Pilet, Davide Frey, and François Taïani. "Simple, Efficient and Convenient Decentralized Multi-Task Learning for Neural Networks." In: 2021.

[BG97] Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling - Theory and Applications*. 1997.

[DG80] Jacques H. Drèze and Joseph Greenberg. "Hedonic Coalitions: Optimality and Stability." In: *Econometrica* 48.4 (1980), pp. 987–1003.

[DH04] Marco F. Duarte and Yu Hen Hu. "Vehicle classification in distributed sensor networks." In: *Journal of Parallel and Distributed Computing* 64.7 (2004), pp. 826–838.

[DV05] Ian Davidson and Sergei Vassilvitskii. "Agglomerative Hierarchical Clustering with Constraints: Theoretical and Empirical Results." In: *Knowledge Discovery in Databases: PKDD 2005*. Vol. 3721. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 59–70.



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

Figure 23: Clusters found in Buddymove@115 (Manifold MDS)

[Est+96] Martin Ester et al. "A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. AAAI Press, 1996, pp. 226–231.

[Fer11] Nuno Gonçalo Costa Fernandes Marques de Abreu. "Análise do perfil do cliente Recheio e desenvolvimento de um sistema promocional." MA thesis. Instituto Universitário de Lisboa, 2011.

[GCT14] Nuwan Ganganath, Chi-Tsun Cheng, and Chi Kong Tse. "Data Clustering with Cluster Size Constraints Using a Modified K-Means Algorithm." In: *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE, 2014, pp. 158–161.

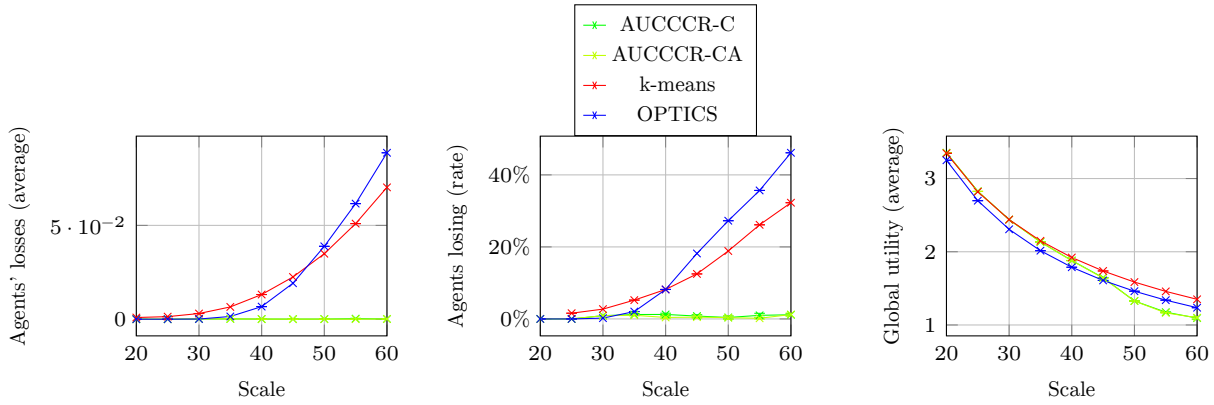
[KT79] Daniel Kahneman and Amos Tversky. "Prospect Theory: An Analysis of Decision under Risk." In: *Econometrica* 47.2 (1979), pp. 263–291.

[Llo82] Stuart P. Lloyd. "Least squares quantization in PCM." In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137.

[Mei+19] Jie Mei et al. "Coalitional Game Theory Based Local Power Exchange Algorithm for Networked Microgrids." In: *Applied Energy* 239 (2019), pp. 133–141.

[Ped+11] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830.

[RA14] Shini Renjith and C Anjali. "A personalized mobile travel recommender system using hybrid algorithm." In: *2014 First International Conference on Computational Systems and Communications (ICCSC)*. IEEE, 2014, pp. 12–17.



(a) Agents' losses caused (b) Share of agents having losses (c) Global utility achieved

Figure 24: Metrics in Wholesale for different scales by different algorithms

[Ran+07] Nagarajan Ranganathan et al. "An Automated Decision Support System Based on Game Theoretic Optimization for Emergency Management in Urban Environments." In: *Journal of Homeland Security and Emergency Management* 4.2 (2007).

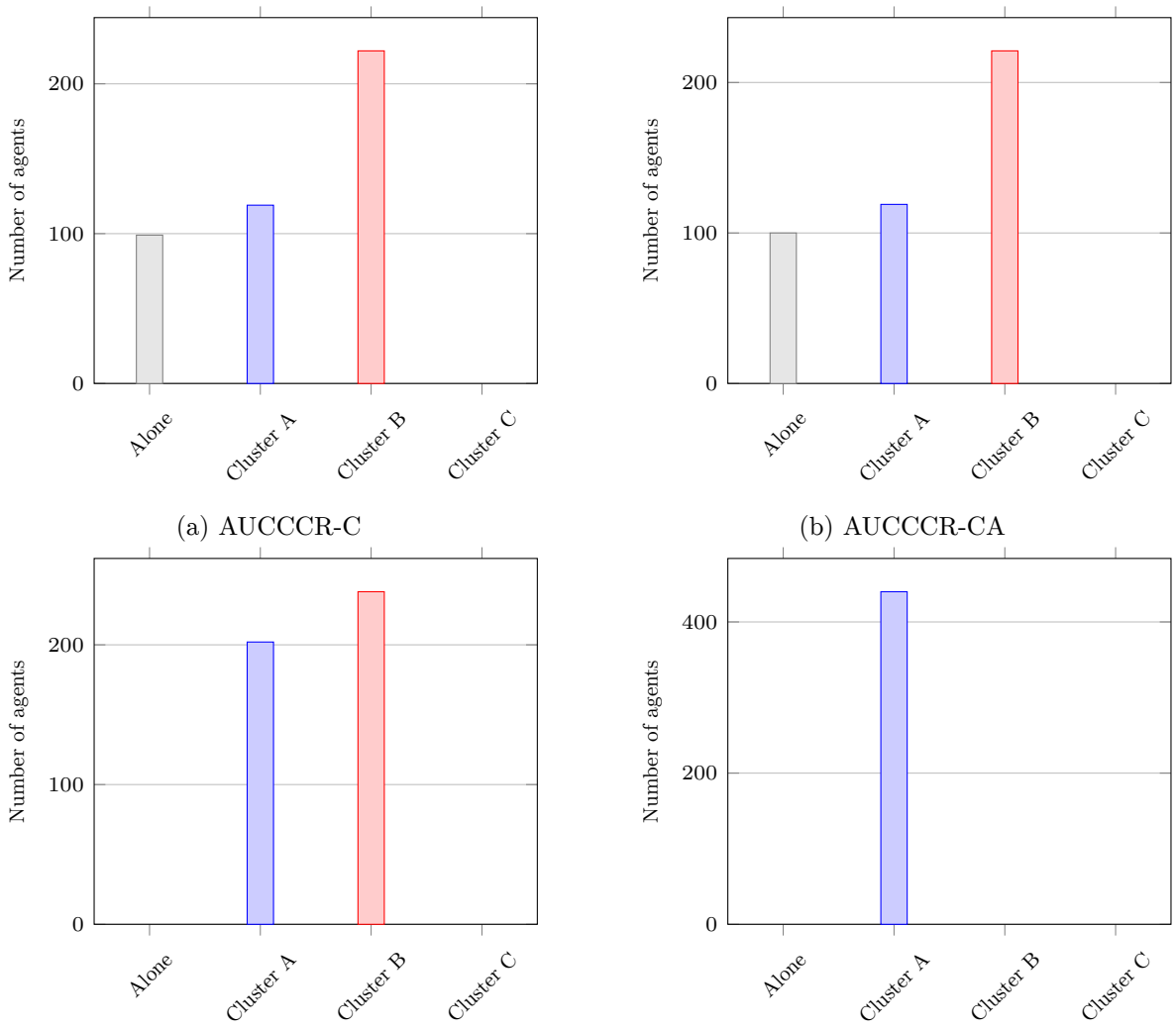
[Saa+10] Walid Saad et al. "Coalition Formation Games for Distributed Cooperation Among Roadside Units in Vehicular Networks." In: *IEEE Journal on Selected Areas in Communications* 29.1 (2010), pp. 48–60.

[SvW19] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. "Learning Task Relatedness in Multi-Task Learning for Images in Context." In: ICMR '19. Association for Computing Machinery, 2019, pp. 78–86.

[Tan+19] Wei Tang et al. "Size Constrained Clustering With MILP Formulation." In: *IEEE Access*. Vol. 8. IEEE, 2019, pp. 1587–1599.

[vM44] John von Neumann and Oskar Morgenstern. *Theory Of Games And Economic Behavior*. 1944.

[ZCY17] Tian Zhang, Wei Chen, and Feng Yang. "Data offloading in mobile edge computing: A coalitional game based pricing approach." In: *IEEE Access* 6 (2017), pp. 2760–2767.



(a) AUCCCR-C

(b) AUCCCR-CA

(c) k-means

(d) OPTICS

Figure 25: Clusters size in Wholesale for scale 45 by different algorithms

Algorithm 3: Clustering algorithm with guaranteed convergence

function clusterConv(A, p, k) **is**

  /* k-means++ initialization */

  $ra \leftarrow rand(A)$

  $ngrp[0] \leftarrow \{ra\}$

  foreach $a \in A$ **do**

    $dmin2[a] \leftarrow d(p(a), p(ra))^2$

  for $1 \leqslant i < k$ **do**

    $na \leftarrow rand(A, dmin2)$

      /* random element from A, relatives probabilities given by $dmin2$ */

    $ngrp[i] \leftarrow \{na\}$ // appends $\{na\}$ to ngrp

    foreach $a \in A$ **do**

      $dmin2[a] \leftarrow \min(d(p(a), p(na))^2, dmin2[a])$

  $nval \leftarrow \#A$

  /* main loop */

  repeat

    $val \leftarrow nval$

    $grp \leftarrow ngrp$

    $ngrp.clear()$

    $nsize \leftarrow [\#A ... \#A]$

    repeat

      $size \leftarrow nsize$

      $nsize \leftarrow [0...0]$

      foreach $a \in A$ **do**

        $bgrp[a] \leftarrow \text{argmax}_{i \in \bot \cup [\![0, k-1]\!]}(n\Big(d(p(a), bary(grp[i] \cup a))\Big) \times v(size[i] + \mathbb{1}_{a \notin grp[i]}))$

        /* assuming $grp[\bot] = \emptyset$ and $size[\bot] = 0$ */

        $nsize[bgrp[a]] + +$ // does nothing if $bgrp[a] = \bot$

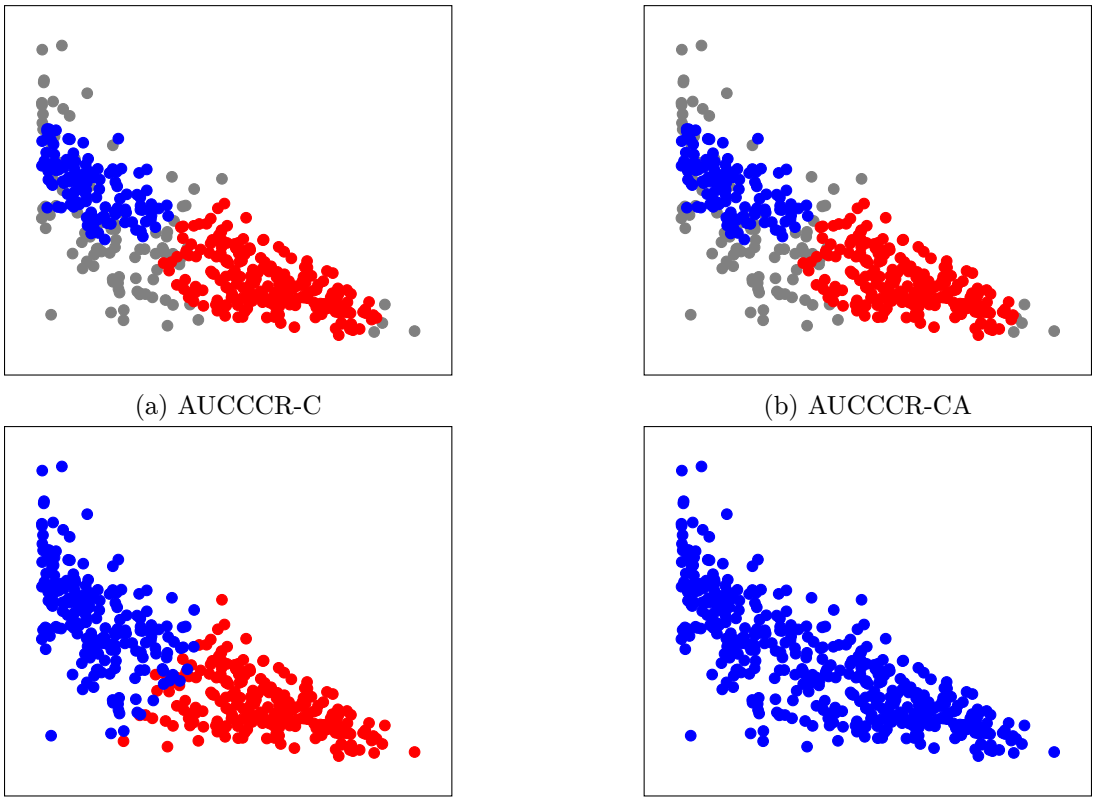
    until $\sum nsize \geqslant \sum size$

    $ngrp \leftarrow bgrp.invertkv()$ // key-value inversion

    $nval \leftarrow \sum_{a \in A} n\Big(d(p(a), bary(ngrp.group(a)))\Big) \times v(\#ngrp.group(a))$

  until $nval \leqslant val$

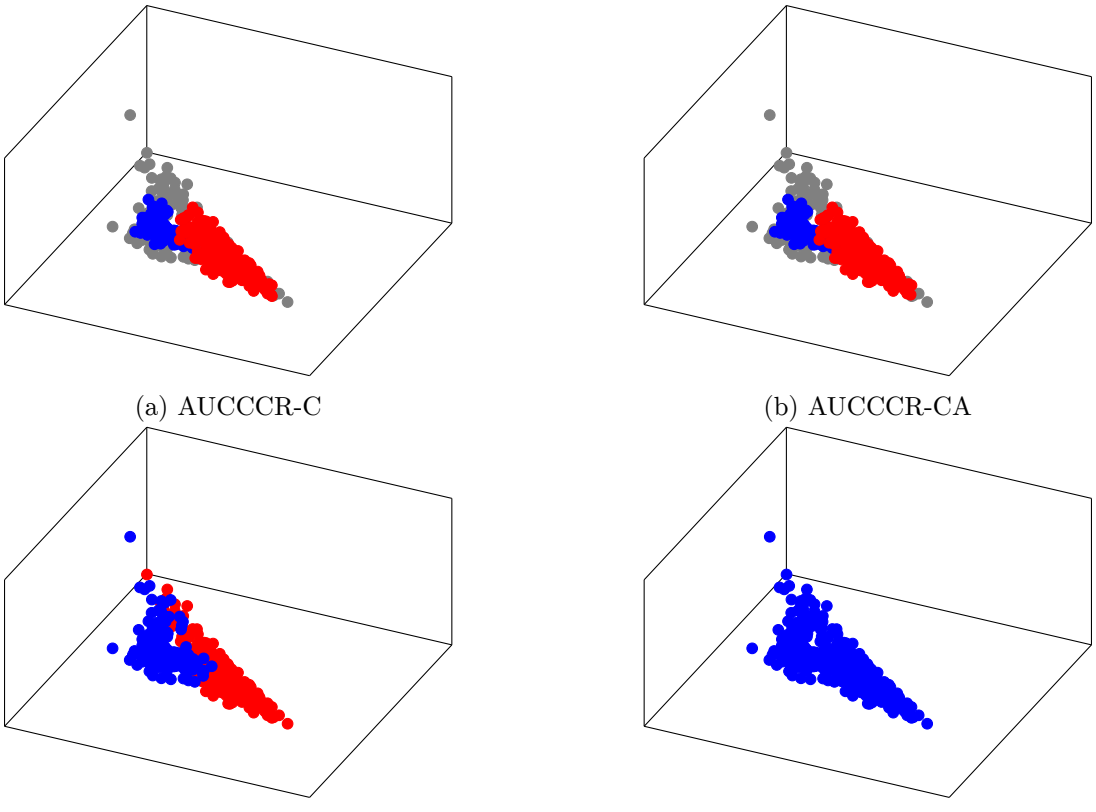
  return grp



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

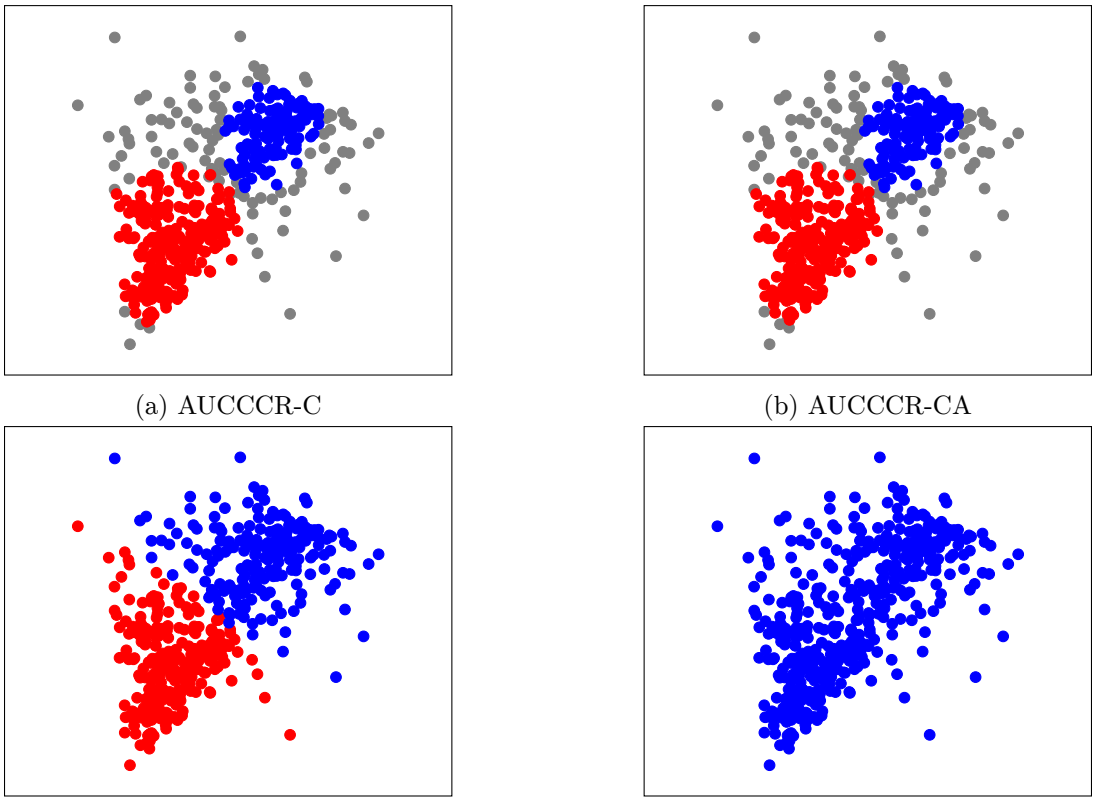
Figure 26: Clusters found in Wholesale@45



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

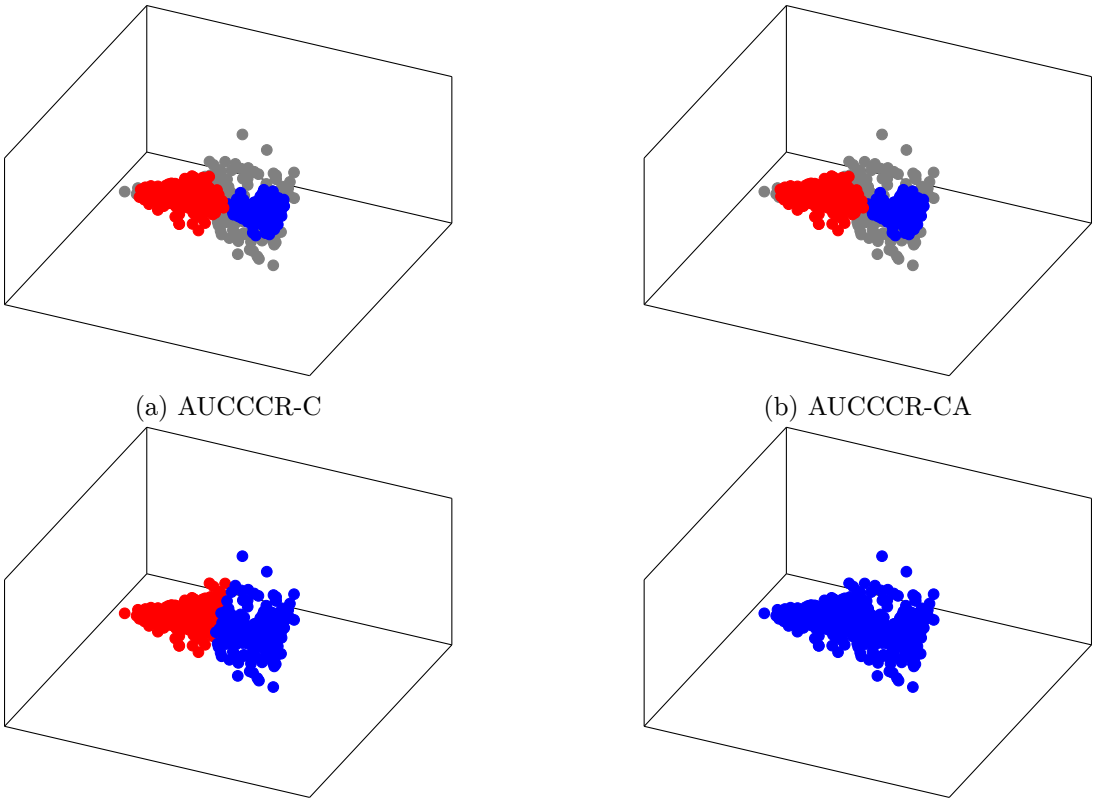
Figure 27: Clusters found in Wholesale@45



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

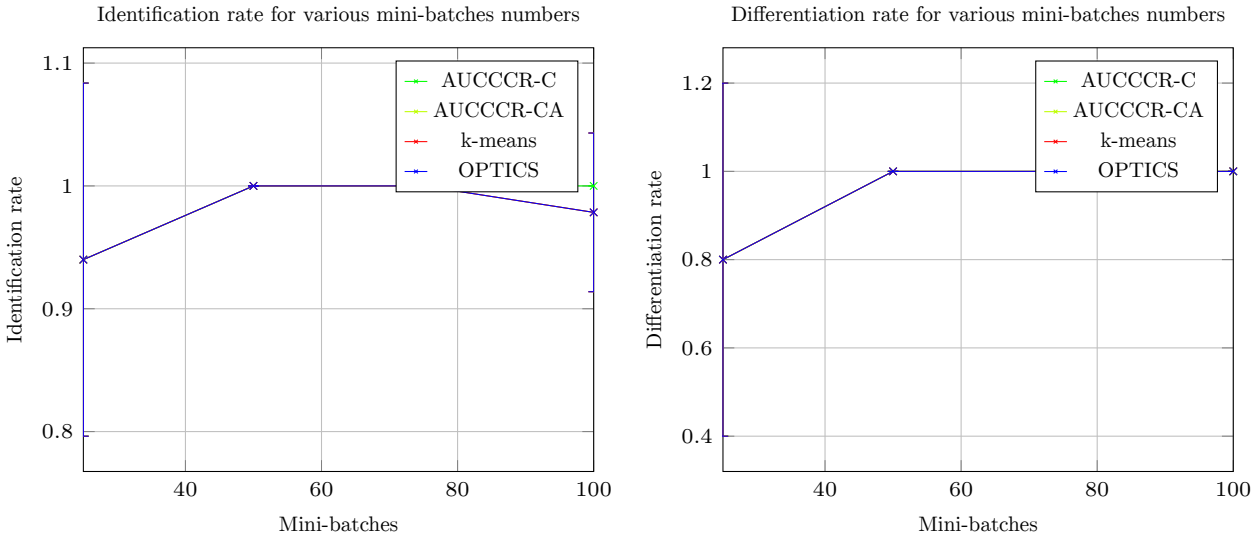
Figure 28: Clusters found in Wholesale@45 (Manifold MDS)



(a) AUCCCR-C (b) AUCCCR-CA

(c) k-means (d) OPTICS

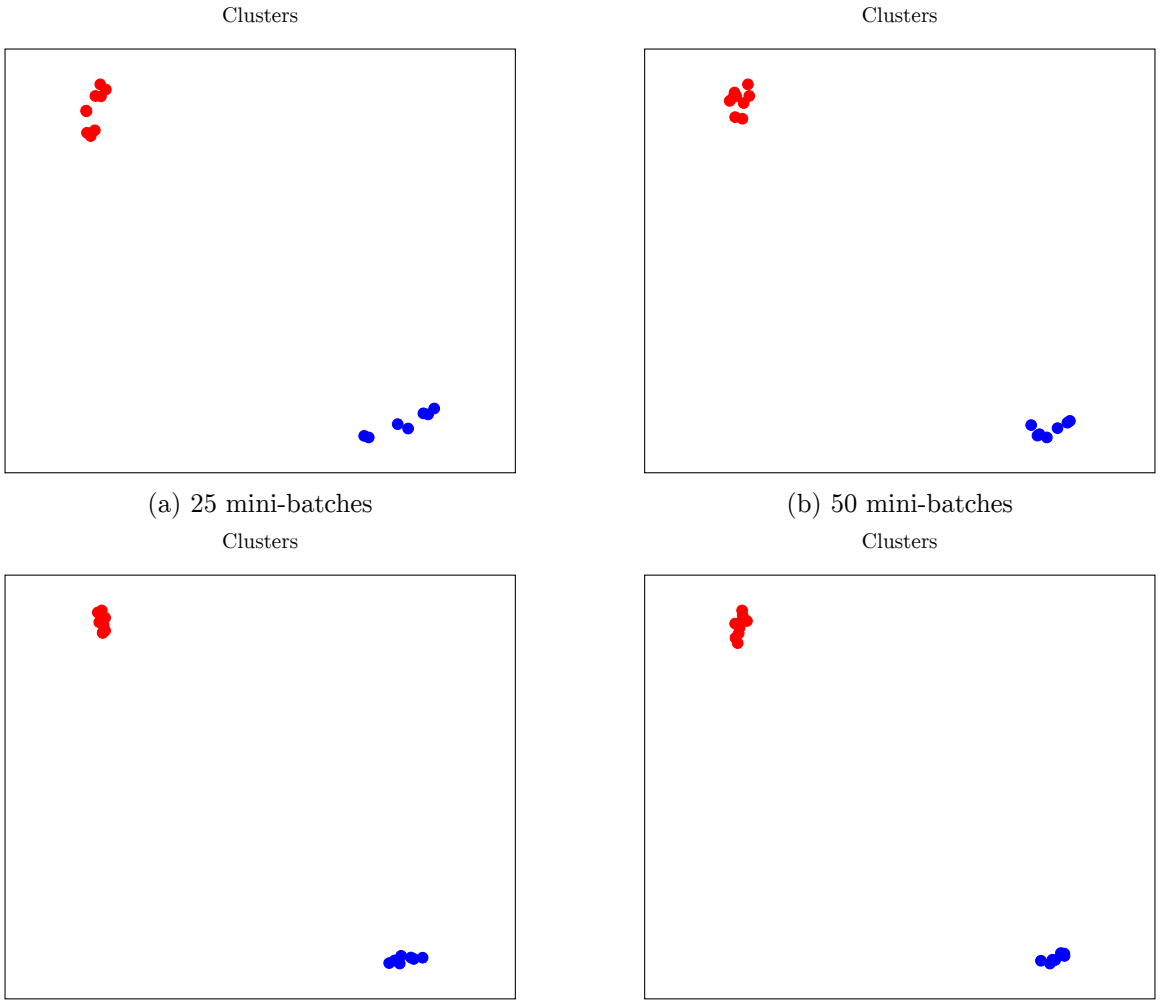
Figure 29: Clusters found in Wholesale@45 (Manifold MDS)



(a) Identification rate

(b) Differentiation rate

Figure 30: Metrics for various mini-batches numbers and different algorithms



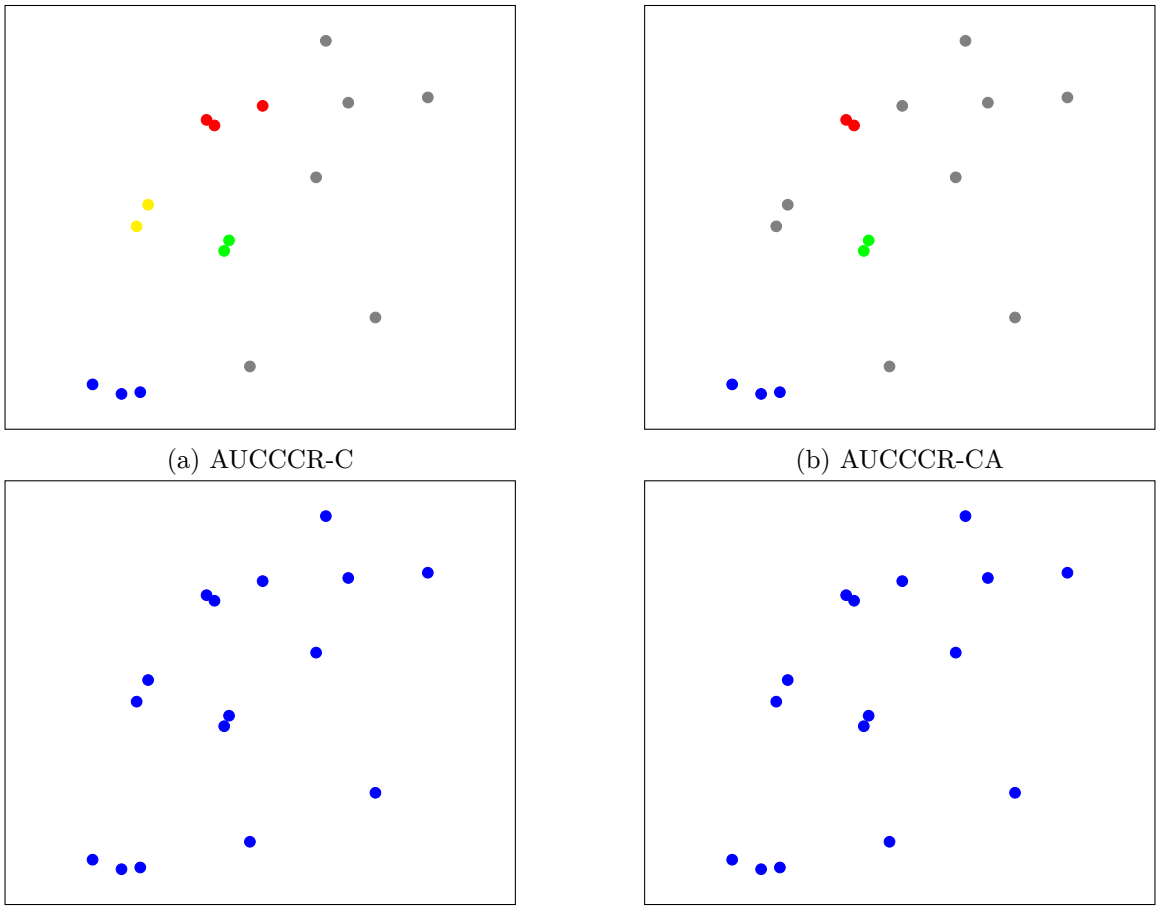
(a) 25 mini-batches

(b) 50 mini-batches

(c) 75 mini-batches

(d) 100 mini-batches

Figure 31: Clusters found by AUCCCR-C in MNIST for various numbers of mini-batches



(a) AUCCCR-C

(b) AUCCCR-CA

(c) k-means

(d) OPTICS

Figure 32: Clusters found by various algorithms in VSN

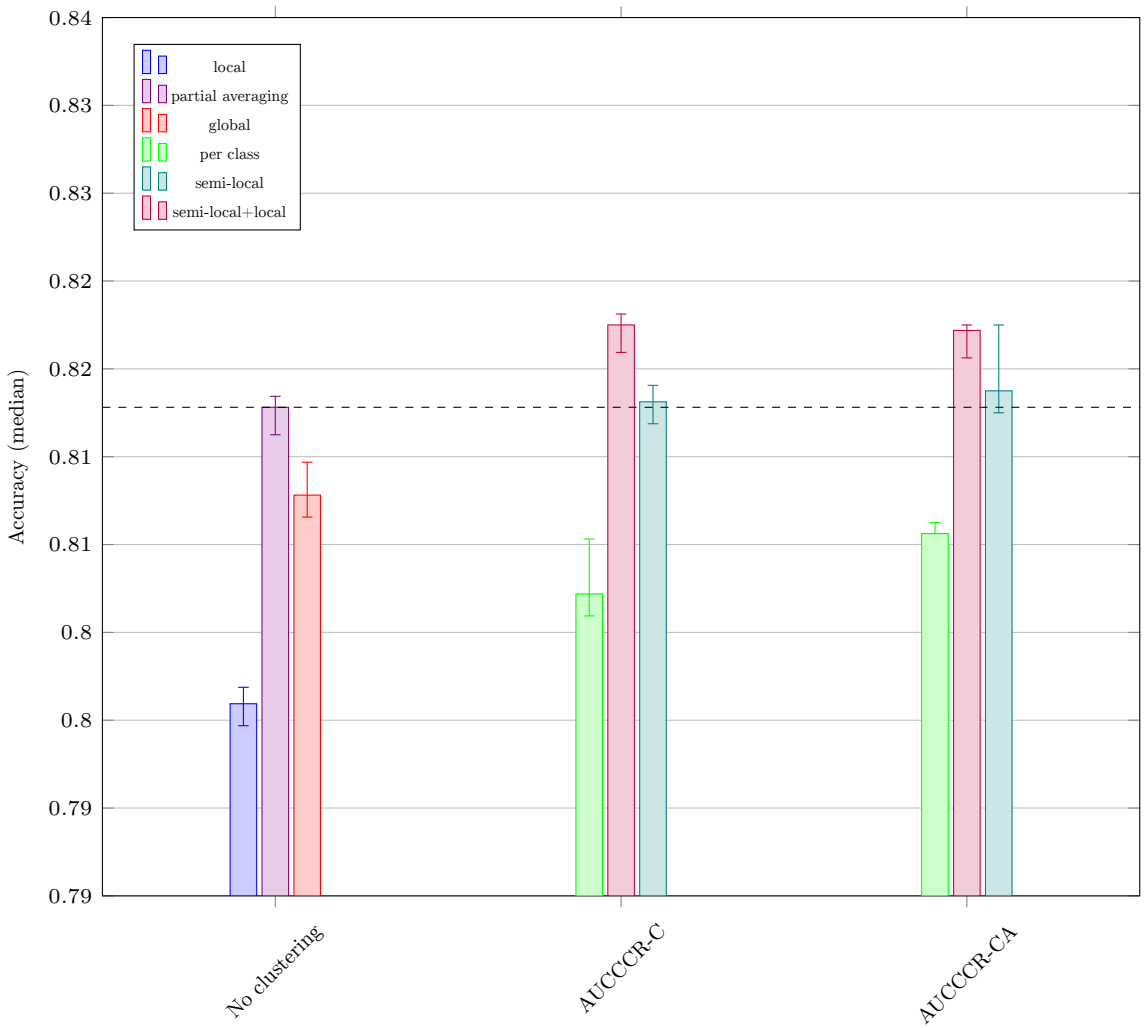


Figure 33: Clustering + federated ML in VSN